REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 29-02-2012	**2. REPORT TYPE** Final performance report	**3. DATES COVERED** *(From - To)* March 2009 -- Feb. 2012

4. TITLE AND SUBTITLE Joint Information Theoretic and Differential Geometrical Approach for Robust Automated Target Recognition	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER FA9550-09-1-0132
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Dapeng Wu	**5d. PROJECT NUMBER** 00075874
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Department of Electrical & Computer Engineering University of Florida Gainesville, Florida 32611-6130	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) USAF, AFRL AF OFFICE OF SCIENTIFIC RESEARCH 875 N. RANDOLPH ST. ROOM 3112 ARLINGTON, VA 22203	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) AFRL-OSR-VA-TR-2012-0357

12. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The overall objective of this project is to develop transformative theory and algorithms for robust Automated Target Recognition (ATR). This project addressed the following challenging problems in ATR: modeling uncertainty, small sample size, high dimensional data, irrelevant features/dimensions, heterogeneous data, and outliers. In this project, the PI proposed and developed the following new techniques:
1) kernel local feature extraction (KLFE) for ATR applications, 2) technique for identifying network dynamics under sparsity and stationarity constraints, 3) self-organized-queue-based (SOQ) clustering scheme, 4) robust principal component analysis (RPCA) based on manifold optimization, outlier detection, and subspace decomposition.

15. SUBJECT TERMS
Feature extraction, clustering, identification of network dynamics, robust principal component analysis

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dapeng Wu
U	U	U	UU	62	**19b. TELEPHONE NUMBER** *(Include area code)* 352-392-4954

Reset

Air Force Office of Scientific Research Grant FA9550-09-1-0132:

Joint Information Theoretic and Differential Geometrical Approach for Robust Automated Target Recognition

(Performance Period: March 2009 – February 2012)

Final Report

PI: Dapeng “Oliver” Wu

Department of Electrical and Computer Engineering

University of Florida

431 Engineering Building, P.O.Box 116130

Gainesville, FL 32611-6130

Tel: (352) 392-4954, Fax: (352) 392-0044

Email: wu@ece.ufl.edu, Web: http://www.wu.ece.ufl.edu

February 29, 2012

Contents

List of Figures

List of Tables

1 Summary

The overall objective of this project is to develop transformative theory and algorithms for robust Automated Target Recognition (ATR). This project addressed the following challenging problems in ATR: modeling uncertainty, small sample size, high dimensional data, irrelevant features/dimensions, heterogeneous data, and outliers. In this project, the PI proposed and developed the following new techniques:

- Kernel-based feature extraction under maximum margin criterion: We developed a technique called kernel local feature extraction (KLFE) for ATR applications. Compared with other feature extraction algorithms, KLFE enjoys nice properties such as low computational complexity, and high probability of identifying relevant features; this is because KLFE is a nonlinear wrapper feature extraction method and consists of solving a simple convex optimization problem. The experimental results have shown the superiority of KLFE over the existing algorithms.
- Identification of network dynamics under sparsity and stationarity constraints: The sparse vector autoregressive model (VAR) is commonly used for modeling dynamic networks, such as tank movements, troops movements, brain functional networks, stock markets and social networks. A penalized linear regression was proposed to identify the autoregressive coefficient matrices with sparsity constraint. However, though the VAR model is assumed to be stationary, this property is never taken into consideration by the penalized linear regression. Moreover, the present techniques for estimating a VAR model are only applicable to the problems with relatively low dimensionality and large number of observations. The main purpose of this work is to tackle these challenging issues. We formulate the problem as penalized linear regression with stationarity constraint, and propose the Berhu iterative sparsity pursuit with stationarity constraint (BIPS) to solve the problem efficiently. Berhu is a novel scheme with hybrid penalty that improves the Lasso scheme for high collinearity problem. We also implement the screening technique into BIPS for dealing with the "large p small n" problem. A bootstrap enhanced learning procedure is applied to approximate the probability of existence for each connection. Experiments show that our method guarantees a stationary estimate, outperforms Lasso in estimation accuracy, and works well for high-dimensional problems.
- Self-organized-queue-based (SOQ) clustering: In this work, we take a bio-inspired approach

to the graph clustering problem and enable fictitious queues with self-organizing capability to cluster nodes in a graph. Our SOQ clustering scheme is one type of kernel clustering schemes. Experimental results have demonstrated the superiority of our SOQ scheme over the existing schemes such as K-means, kernel K-means, spectral clustering and normalized cuts.

- Robust principal component analysis (RPCA): Our RPCA scheme is based on manifold optimization, outlier detection, and subspace decomposition. Experimental results have shown that our RPCA scheme significantly outperforms PCA when there are outliers in the data; our RPCA scheme also performs much better than the existing robust PCA schemes under the condition that both Gaussian noise and outliers exist in the data.

Publications during the reporting period:

- J. Wang, J. Fan, H. Li, D. Wu, "Kernel-based Feature Extraction under Maximum Margin Criterion," Journal of Visual Communication and Image Representation, vol. 23, no. 1, pp. 53–62, January 2012.
- Y. He, Y. She, D. Wu, "Identification of Stationary Sparse Vector Autoregressive Model," to be submitted to Journal of Machine Learning Research.
- Y. She, S. Li, D. Wu, "Manifold-Optimization-Based Robust Principal Component Analysis," to be submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence.
- B. Sun, D. Wu, "Self-Organized-Queue-Based Clustering: a Bio-inspired Approach," to be submitted to Nature.

2 Kernel-based Feature Extraction under Maximum Margin Criterion

In this work, we study the problem of feature extraction for pattern classification applications. RELIEF is considered as one of the best-performed algorithms for assessing the quality of features for pattern classification. Its extension, local feature extraction (LFE), was proposed recently and was shown to outperform RELIEF. In this paper, we extend LFE to the nonlinear case, and develop a new algorithm called kernel LFE (KLFE). Compared with other feature extraction algorithms,

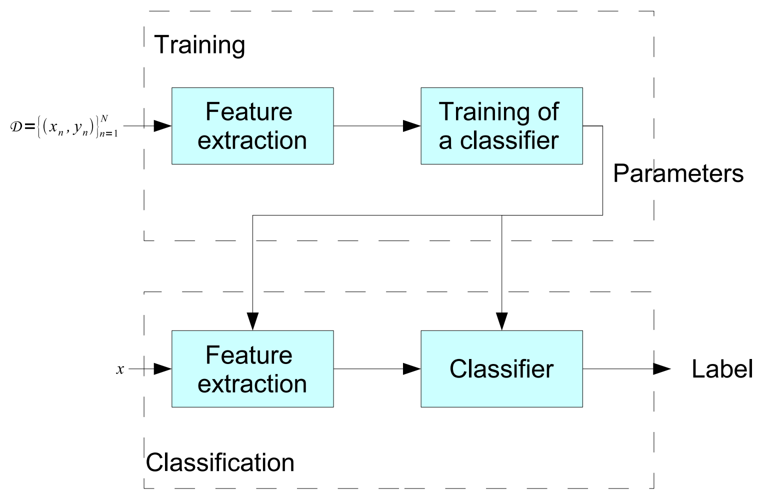


Figure 1: A typical pattern classification system.

KLFE enjoys nice properties such as low computational complexity, and high probability of identifying relevant features; this is because KLFE is a nonlinear wrapper feature extraction method and consists of solving a simple convex optimization problem. The experimental results have shown the superiority of KLFE over the existing algorithms. Next, we present the technical details.

2.1 Introduction

In this paper, we study the problem of feature extraction for pattern classification applications. As shown in Fig. 1, a typical pattern classification system consists of two parts: one for the training phase and one for the classification phase. In the training phase, the system is given a training data set,

$$\mathcal{D} \triangleq \{(\mathbf{x}_n, y_n)\}_{n=1}^N \subset \mathcal{X} \times \mathcal{Y}, \tag{1}$$

where N is the number of samples in the training data set, $\mathcal{X} \subset \mathbb{R}^I$ is the I-dimensional feature space, and $\mathcal{Y} = \{\pm 1\}$ [1] is the label space. At the end of the training phase, the system obtains a parameter set, which provides information needed by the feature extraction module and the classifier in the classification phase. Although in many papers, the feature extraction module for a classification system is not explicitly specified, it is a significant component. A good feature extraction technique not only reduces system complexity and processing time, but also improves classification accuracy by eliminating irrelevant features.

[1]In this paper, we focus on two-category pattern classification problem.

In this paper, we propose a feature extraction algorithm, called kernel local feature extraction (KLFE), which has low computational complexity, and achieves high probability of identifying irrelevant features for removal (dimension reduction).

To begin with, we define feature extraction as a mapping

$$f : \mathcal{X} \subset \mathbb{R}^{I} \to \mathcal{X}' \subset \mathbb{R}^{I'}, \tag{2}$$

which maps patterns in input feature space $\mathcal{X}$ to output feature space $\mathcal{X}'$, in order to optimize some pre-defined criterion. Usually, we have $I' \leqslant I$ so that features are mapped from a high-dimensional space to a lower one, which reduces system complexity.

A feature extraction method can be categorized according to the following criteria.

First, a feature extraction method can be linear or nonlinear. A linear feature extraction method has the form

$$f(\mathbf{x}) = \mathcal{A} \cdot \mathbf{x}, \mathbf{x} \in \mathcal{X}, \tag{3}$$

where $\mathcal{A}$ is a $I' \times I$ matrix. Otherwise, it is nonlinear. Usually nonlinear methods outperform linear ones as nonlinear methods are able to capture the true pattern, which is usually nonlinear. In this paper, we propose a nonlinear feature extraction method, called KLFE, which has the form of

$$f(\mathbf{x}) = \mathcal{A} \cdot \varphi(\mathbf{x}), \tag{4}$$

where $\varphi : \mathcal{X} \subset \mathbb{R}^{I} \to \bar{\mathcal{X}} \subset \mathbb{R}^{\bar{I}}$ is a nonlinear function and $\mathcal{A}$ is an $I' \times \bar{I}$ matrix.

Second, feature extraction has two special cases, namely, *feature selection* and *feature weighting*. If $\mathcal{A}$ in Eqs. (3) and (4) is a diagonal matrix whose diagonal elements are restricted to either 0 or 1, the feature extraction method is also called *feature selection*; if the diagonal elements of diagonal matrix $\mathcal{A}$ can take any real-valued number between 0 and 1, the feature extraction method is also called *feature weighting*.

Allowing diagonal elements of $\mathcal{A}$ to take real-valued numbers, instead of binary ones, enables feature weighting to employ some well-established optimization techniques and thus allows for more efficient algorithm implementation than feature selection. A celebrated feature weighting method is the RELIEF algorithm [1].

One major shortcoming of feature selection and feature weighting is their inability to remove correlation among different feature dimensions so as to achieve a sparser representation of data samples [2]. In some applications, such as object recognition, where there is no need to preserve the physical meaning of individual features, feature extraction is more appropriate than feature selection and feature weighting. For example, Sun and Wu [3] [4] proposed a linear feature extraction method, called local feature extraction (LFE), which outperforms feature selection and feature weighting. In this paper, we extend LFE to a nonlinear one called Kernel LFE.

Third, a feature extraction method can be further categorized as a wrapper method or a filter method [5]. A wrapper method determines the mapping in Eq. (2) by minimizing the classification error rate of a classifier, whereas a filter method does not. Therefore, filter methods are computationally more efficient but usually perform worse than wrapper methods. The famous principal component analysis (PCA) [6, page 115] is a filter method, whereas RELIEF [1] and LFE [3] are wrapper methods, as they both optimize a 1-nearest-neighbor (1-NN) classifier [6, page 174]. Our KLFE algorithm is also a wrapper method in that it optimizes a 1-NN classifier in the nonlinear-transformed space.

Last, a feature extraction method can be obtained by solving a convex optimization problem or a non-convex optimization problem. A convex optimization problem formulation is preferred since it can be solved efficiently, compared to a non-convex optimization problem. PCA, RELIEF, and LFE are all based on convex optimization formulation. Our KLFE algorithm is also based on a convex optimization formulation, which admits a closed-form solution.

In summary, in this paper, we propose a nonlinear wrapper feature extraction method, which is based on a convex optimization formulation.

The remainder of the section is organized as follows. In Section 2.2, related work is briefly reviewed. In Section 2.3, we describe two existing feature extraction techniques, namely, RELIEF and LFE; our proposed KLFE is a generalization of these two algorithms. In Section 2.4, we present a novel feature extraction algorithm, called KLFE. Section 3.5 presents experimental results, which demonstrate that KLFE outperforms the existing feature extraction algorithms. Section 3.7 concludes this work.

2.2 Related Work

In this section, we briefly review some feature extraction algorithms, which will be compared to our KLFE algorithm. Principal component analysis (PCA) [6, page 115] is probably one of the most commonly used algorithms for feature extraction. One major drawback of PCA, however, is that it is targeted at minimizing mean squared error for data compression or efficient data representation, rather than minimizing the classification error probability for pattern classification. Other PCA-type algorithms, e.g., kernel PCA (KPCA) [7], usually perform better than PCA in representing nonlinear relationship among different feature dimensions, but suffer from the same limitation as PCA since they do not use class labels in the training phase; i.e., they are unsupervised algorithms. The KLFE algorithm proposed in this paper, like its predecessors LFE and RELIEF, utilizes the class label information in the training phase; i.e., KLFE is an supervised algorithm.

Among the existing feature weighting techniques, RELIEF [1] is considered as one of the best-performed ones due to its simplicity and effectiveness [8]. RELIEF determines the parameters of diagonal matrix $\mathcal{A}$ in Eq. (3) by solving a convex optimization problem, which maximizes a margin-based criterion [9]. The LFE algorithm, proposed in Ref. [3], is an extension to RELIEF; LFE removes the constraint of matrix $\mathcal{A}$ in Eq. (3) being diagonal, which is required by RELIEF. Both LFE and RELIEF are linear methods. In contrast, our KLFE method is a nonlinear extension to LFE; experimental results show that KLFE performs better than LFE.

In Ref. [10], the authors extended RELIEF to a kernel space, which is the space that contains the image of the nonlinear-transformation used in a kernel method; their approach is to identify an orthonormal basis of the kernel space and perform RELIEF in this kernel space; the resulting schemes are called Feature Space KPCA (FSKPCA) and Feature Space Kernel Gram-Schmidt Process (FSKGP). They showed that FSKPCA and FSKGP achieve similar performance to that of the state-of-the-art algorithms. Our KLFE adopts a similar strategy, i.e., our KLFE first computes an orthonormal basis of the kernel space and then performs LFE in the kernel space. Since LFE achieves improved performance over RELIEF, KLFE is expected to outperform FSKPCA and FSKGP, which are kernel-based versions of RELIEF.

In the next section, we briefly review RELIEF and LFE before we present our KLFE in Section 2.4.

2.3 Review of RELIEF and LFE

In this section, we briefly review RELIEF and LFE since LFE is an extension of RELIEF and KLFE is an extension of LFE. We first define two terms, nearest hit (NH) and nearest miss (NM) in Section 2.3.1, which will be used in all of the three algorithms, i.e., RELIEF, LFE, and KLFE. Then we introduce RELIEF and LFE in Sections 2.3.2 and 2.3.3, respectively.

2.3.1 Nearest Hit (NH) and Nearest Miss (NM)

Suppose we are given a training data set, as shown in Eq. (1). For any pattern $(\mathbf{x}, y) \in \mathcal{D}$, we define its nearest hit (NH) as

$$NH(\mathbf{x}, y) \overset{\triangle}{=} \arg\min_{\mathbf{x}'} ||\mathbf{x}' - \mathbf{x}||_p \quad (5)$$

$$s.t.\ (\mathbf{x}', y') \in \mathcal{D}, \quad (6)$$

$$y' = y, \quad (7)$$

and its nearest miss (NM) as

$$NM(\mathbf{x}, y) \overset{\triangle}{=} \arg\min_{\mathbf{x}'} ||\mathbf{x}' - \mathbf{x}||_p \quad (8)$$

$$s.t.\ (\mathbf{x}', y') \in \mathcal{D}, \quad (9)$$

$$y' \neq y, \quad (10)$$

where $||\mathbf{x}||_p$ is L^p-norm of vector $\mathbf{x}$. In this paper, we let $p = 1$, i.e., we choose L^1 norm.

Using Eqs. (5) and (8), we further denote

$$\mathbf{m}_n \overset{\triangle}{=} \mathbf{x}_n - NM(\mathbf{x}_n, y_n), \quad (11)$$

$$\mathbf{h}_n \overset{\triangle}{=} \mathbf{x}_n - NH(\mathbf{x}_n, y_n), \quad (12)$$

for $n = 1, \ldots, N$.

2.3.2 RELIEF

Now we briefly introduce RELIEF. Denote by $\mathbf{w} = [w_1, \ldots, w_I]^T$ the weight vector, where w_i is the weight of the i-th dimension of $\mathbf{x}_n \in \mathbb{R}^I$. RELIEF defines the margin of a pattern $(\mathbf{x}_n, y_n) \in \mathcal{D}$ as

$$\rho_n \overset{\triangle}{=} ||\mathbf{m}_n||_1 - ||\mathbf{h}_n||_1,\ n = 1, \ldots, N. \quad (13)$$

Then the objective of RELIEF is to maximize the overall margin over weight vector, i.e.,

$$\max_{\mathbf{w}} \quad \textstyle\sum_{n=1}^N \rho_n(\mathbf{w}) = \sum_{n=1}^N \left(\mathbf{w}^T |\mathbf{m}_n| - \mathbf{w}^T |\mathbf{h}_n|\right), \quad (14a)$$

$$\text{s.t.} \quad ||\mathbf{w}||_2^2 = 1, \mathbf{w} \geqslant 0, \quad (14b)$$

where $|\cdot|$ denotes element-wise absolute operator. Let

$$\mathbf{z} \triangleq \sum_{n=1}^{N} \left(|\mathbf{m}_n| - |\mathbf{h}_n|\right), \tag{15}$$

we simplify Eq. (14) to

$$\max_{\mathbf{w}} \quad \mathbf{w}^T z, \text{ s.t. } \|\mathbf{w}\|_2^2 = 1, \mathbf{w} \geqslant 0. \tag{16}$$

Applying Lagrangian multipliers λ and θ to Eq. (16), one obtains

$$L = \quad -\mathbf{w}^T z + \lambda \left(\|\mathbf{w}\|_2^2 - 1\right) + \mathbf{w}^T \theta. \tag{17}$$

Taking derivative with respect to $\mathbf{w}$ at both sides of Eq. (17) and setting it to zero results in

$$\begin{aligned} \tfrac{\partial L}{\partial \mathbf{w}} &= -\mathbf{z} + 2\lambda \mathbf{w} - \theta = 0 \\ \Rightarrow \qquad \mathbf{w} &= \tfrac{1}{2\lambda}(\mathbf{z} + \theta). \end{aligned} \tag{18}$$

The closed-form solution to Eq. (14) is [9]

$$\mathbf{w} = \frac{(\mathbf{z})^+}{\|(\mathbf{z})^+\|_2}, \tag{19}$$

where $(\mathbf{z})^+ = [(z_1)^+, \ldots, (z_I)^+]^T$ and $(\cdot)^+ = \max(\cdot, 0)$. The RELIEF algorithm specifies the projection matrix

$$\mathcal{A} = \begin{bmatrix} w_1 & & 0 \\ & \ddots & \\ 0 & & w_I \end{bmatrix}.$$

2.3.3 LFE

A natural extension of RELIEF is to use a full matrix instead of a diagonal matrix, which results in LFE [3]. In LFE, the following optimization problem is considered:

$$\max_{\mathbf{W}} \quad \sum_{n=1}^{N} \rho_n(\mathbf{W}) = \sum_{n=1}^{N} \mathbf{m}_n^T \mathbf{W} \mathbf{m}_n - \sum_{n=1}^{N} \mathbf{h}_n^T \mathbf{W} \mathbf{h}_n, \tag{20a}$$

$$\text{s.t.} \qquad \|\mathbf{W}\|_F^2 = 1, \mathbf{W} \geqslant 0, \tag{20b}$$

where $\|\mathbf{W}\|_F$ is the Frobenius norm of $\mathbf{W}$, i.e., $\|\mathbf{W}\|_F = \sqrt{\sum_{i,j} w_{i,j}^2} = \sqrt{\sum_i \lambda_i^2}$, where $\{\lambda_i\}_{i=1}^{I}$ are the eigenvalues of $\mathbf{W}$.

Sun and Wu [3] proved Theorem 1, which provides a solution to Eq. (20).

Theorem 1. *Let*

$$\Sigma_{mh} \triangleq \sum_{n=1}^{N} \mathbf{m}_n \mathbf{m}_n^T - \sum_{n=1}^{N} \mathbf{h}_n \mathbf{h}_n^T, \tag{21}$$

and let $\{(\sigma_i, \mathbf{a}_i)\}_{i=1}^{I}$ *be the eigen-system of* Σ_{mh} *such that* $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_I$. *The solution for Eq.* (20)*, up to the difference of a constant, is*

$$\mathbf{W} = \sum_{\{i:\sigma_i>0\}} \sigma_i \mathbf{a}_i \mathbf{a}_i^T. \tag{22}$$

According to Theorem 1, the LFE algorithm produces a projection matrix by maintaining the dimensions specified by eigenvectors $\{\mathbf{a}_i\}_{i=1}^{I'}$, which correspond to the largest I' eigenvalues of Σ_{mh} where $I' \leqslant I$ is the target dimension size as defined in Eq. (3) and I' should be chosen such that $\sigma_1 \geqslant \cdots \geqslant \sigma_{I'} \geqslant 0$; in other words, LFE is defined by

$$f(\mathbf{x}) = \mathcal{A}\mathbf{x}, \ \mathbf{x} \in \mathcal{X}, \tag{23}$$

where

$$\mathcal{A} = \left[\sqrt{\sigma_1}\mathbf{a}_1, \ldots, \sqrt{\sigma_{I'}}\mathbf{a}_{I'}\right]^T. \tag{24}$$

2.4 Kernel LFE

In this section, we propose KLFE algorithm. This section is organized as follows. In Section 2.4.1, we present the extension of LFE to a high-dimensional space by introducing a nonlinear mapping. In Section 2.4.2, we prove that both LFE and KLFE are basis rotation invariant and thus KLFE can be considered to perform LFE under an orthonormal basis in the kernel space. In Section 2.4.3, we present an algorithm for implementing KLFE by using KPCA to find an orthonormal basis in the kernel space. We summarize KLFE algorithm and describe the implementation details in Section 2.4.4. Computational complexity is also analyzed in this section.

2.4.1 Nonlinear LFE in High-Dimensional Space

As presented in Section 2.3.3, the LFE algorithm is a linear feature extraction method as in Eq. (3). Our idea of extending LFE is to introduce a nonlinear function,

$$\varphi : \mathcal{X} \subset \mathbb{R}^I \rightarrow \bar{\mathcal{X}} \subset \mathbb{R}^{\bar{I}}, \tag{25}$$

where usually $\bar{I} \gg I$, which maps patterns from a low-dimensional space to a high-dimensional one. We call $\bar{\mathcal{X}}$ or $\mathbb{R}^{\bar{I}}$ *kernel space*, which contains the image of φ. Then we apply the LFE algorithm to kernel space $\bar{\mathcal{X}}$; the resulting algorithm is called KLFE. We further assume $\bar{I} \gg N$, as is always the case when using a kernel method. Similar to Eqs. (11) and (12), we let

$$\bar{\mathbf{m}}_n \triangleq \varphi(\mathbf{x}_n) - \varphi\left(NM(\mathbf{x}_n, y_n)\right), \tag{26}$$

$$\bar{\mathbf{h}}_n \triangleq \varphi(\mathbf{x}_n) - \varphi\left(NH(\mathbf{x}_n, y_n)\right), \tag{27}$$

$n = 1, \ldots, N$. KLFE can be obtained by solving the following optimization problem,

$$\max_{\bar{\mathbf{W}}} \ \textstyle\sum_{n=1}^{N} \bar{\mathbf{m}}_n^T \bar{\mathbf{W}} \bar{\mathbf{m}}_n - \sum_{n=1}^{N} \bar{\mathbf{h}}_n^T \bar{\mathbf{W}} \bar{\mathbf{h}}_n, \tag{28a}$$

$$\text{s.t.} \quad \|\bar{\mathbf{W}}\|_F^2 = 1, \bar{\mathbf{W}} \geqslant 0. \tag{28b}$$

Since the only difference between Eq. (28) and Eq. (20) is the use of mapping φ, one can directly use Theorem 1 to solve Eq. (28). Corollary 1.1 summarizes the solution to Eq. (28).

Corollary 1.1. *Let*

$$\bar{\Sigma}_{mh} \triangleq \textstyle\sum_{n=1}^{N} \bar{\mathbf{m}}_n \bar{\mathbf{m}}_n^T - \sum_{n=1}^{N} \bar{\mathbf{h}}_n \bar{\mathbf{h}}_n^T, \tag{29}$$

and let $\{(\bar{\sigma}_i, \bar{\mathbf{a}}_i)\}_{i=1}^{\bar{I}}$ be the eigen-system of $\bar{\Sigma}_{mh}$, such that $\bar{\sigma}_1 \geqslant \bar{\sigma}_2 \geqslant \cdots \geqslant \bar{\sigma}_{\bar{I}}$. The solution to Eq. (28)*, up to the difference of a constant, is*

$$\bar{\mathbf{W}} = \sum_{\{i:\bar{\sigma}_i > 0\}} \bar{\sigma}_i \bar{\mathbf{a}}_i \bar{\mathbf{a}}_i^T.$$

From Corollary 1.1, the KLFE algorithm produces a projection matrix by maintaining the dimensions specified by eigenvectors $\{\bar{\sigma}_i\}_{i=1}^{I'}$, which correspond to the largest I' eigenvalues of $\bar{\Sigma}_{mh}$ where $I' \leqslant \bar{I}$ is the target dimension size as defined in Eq. (3) and I' should be chosen such that $\bar{\sigma}_1 \geqslant \cdots \geqslant \bar{\sigma}_{I'} \geqslant 0$; in other words, KLFE is defined by

$$f(\mathbf{x}) = \bar{\mathcal{A}} \varphi\left(\mathbf{x}\right), \mathbf{x} \in \mathcal{X}, \tag{30}$$

where

$$\bar{\mathcal{A}} = \left[\sqrt{\bar{\sigma}_1} \bar{\mathbf{a}}_1, \ldots, \sqrt{\bar{\sigma}_{I'}} \bar{\mathbf{a}}_{I'}\right]^T. \tag{31}$$

2.4.2 Basis Rotation Invariant Property of KLFE

In this section, we study an important property of KLFE: basis rotation invariance. Before we show the basis rotation invariant property of KLFE in Proposition 1, we present Lemma 1.

Lemma 1 states that LFE is basis rotation invariant.

Lemma 1. *Let $\{\mathbf{e}_1^{(i)}\}_{i=1}^I$ and $\{\mathbf{e}_2^{(i)}\}_{i=1}^I$ be two different orthonormal bases in input feature space $\mathbb{R}^I$. Assume that $\{\mathbf{e}_2^{(i)}\}_{i=1}^I$ can be obtained by counterclockwise rotating $\{\mathbf{e}_1^{(i)}\}_{i=1}^I$, and the rotation matrix is denoted by Q. Then, LFE is basis rotation invariant, i.e., a feature vector extracted by LFE under $\{\mathbf{e}_1^{(i)}\}_{i=1}^I$ is the same as the feature vector extracted by LFE under $\{\mathbf{e}_2^{(i)}\}_{i=1}^I$.*

Proof. Assume training samples are given under basis $\{\mathbf{e}_1^{(i)}\}_{i=1}^I$, i.e.,

$$\mathcal{D} \triangleq \{(\mathbf{x}_n, y_n)\}_{n=1}^N \subset \mathcal{X} \times \mathcal{Y},$$

where $\mathcal{X} \subset \mathbb{R}^I$ is the I-dimensional feature space and $\mathcal{Y} = \{\pm 1\}$. Under basis $\{\mathbf{e}_1^{(i)}\}_{i=1}^I$, LFE is formulated as follows:

$$\max_{\mathbf{W_1}} \quad \textstyle\sum_{n=1}^N \rho_n(\mathbf{W_1}) = \sum_{n=1}^N \mathbf{m}_n^T \mathbf{W_1} \mathbf{m}_n - \sum_{n=1}^N \mathbf{h}_n^T \mathbf{W_1} \mathbf{h}_n, \tag{32a}$$

$$\text{s.t.} \quad \|\mathbf{W_1}\|_F^2 = 1, \mathbf{W_1} \geqslant 0. \tag{32b}$$

From Theorem 1, the projection matrix of LFE under basis $\{\mathbf{e}_1^{(i)}\}_{i=1}^I$ is given by

$$\mathcal{A}_1 = \left[\sqrt{\sigma_1}\mathbf{a}_1, \ldots, \sqrt{\sigma_{I'}}\mathbf{a}_{I'}\right]^T, \tag{33}$$

where $I' \leqslant I$ is the target dimension size and $\{\mathbf{a}_i\}$ are the eigenvectors of Σ_{mh} corresponding to the largest I' eigenvalues.

Under basis $\{\mathbf{e}_2^{(i)}\}_{i=1}^I$, the training samples become

$$\mathcal{D} \triangleq \{(Q\mathbf{x}_n, y_n)\}$$

The nearest miss and nearest hit remain the same for each data sample since the Euclidean distance does not change under different orthonormal bases. Under basis $\{\mathbf{e}_2^{(i)}\}_{i=1}^I$, LFE is formulated as below:

$$\max_{\mathbf{W_2}} \quad \sum_{n=1}^{N} \rho_n(\mathbf{W_2}) = \sum_{n=1}^{N}(Q\mathbf{m}_n)^T\mathbf{W_2}Q\mathbf{m}_n - \sum_{n=1}^{N}(Q\mathbf{h}_n)^T\mathbf{W_2}Q\mathbf{h}_n, \tag{34a}$$

$$\text{s.t.} \quad \|\mathbf{W_2}\|_F^2 = 1, \mathbf{W_2} \geqslant 0. \tag{34b}$$

Comparing (32) and (34), we have $\mathbf{W_1} = Q^T\mathbf{W_2}Q$. Then we have

$$\mathbf{W_2} = Q\mathbf{W_1}Q^T \tag{35}$$

$$\overset{(a)}{=} Q(\sum_{\{i:\sigma_i>0\}} \sigma_i \mathbf{a}_i \mathbf{a}_i^T)Q^T \tag{36}$$

$$= \sum_{\{i:\sigma_i>0\}} \sigma_i Q \mathbf{a}_i \mathbf{a}_i^T Q^T \tag{37}$$

where (a) is due to (22). Let the projection matrix of LFE under basis $\{\mathbf{e}_2^{(i)}\}_{i=1}^{I}$, be $\mathcal{A}_2$. Then, from (37), (22), and (24), we have

$$\mathcal{A}_2 = [\sqrt{\sigma_1}Q\mathbf{a}_1, \ldots, \sqrt{\sigma_{I'}}Q\mathbf{a}_{I'}]^T \tag{38}$$

$$\overset{(a)}{=} \mathcal{A}_1 Q^T \tag{39}$$

where (a) is due to (33). Then, the feature vector extracted by LFE under $\{\mathbf{e}_2^{(i)}\}_{i=1}^{I}$ is given by

$$\mathcal{A}_2 Q\mathbf{x} \overset{(a)}{=} \mathcal{A}_1 Q^T Q\mathbf{x} \tag{40}$$

$$\overset{(b)}{=} \mathcal{A}_1\mathbf{x} \tag{41}$$

where (a) is due to (39); (b) is due to the fact that Q is an orthogonal matrix. Eq. (41) means that a feature vector extracted by LFE under $\{\mathbf{e}_1^{(i)}\}_{i=1}^{I}$, i.e., $\mathcal{A}_1\mathbf{x}$ is the same as the feature vector extracted by LFE under $\{\mathbf{e}_2^{(i)}\}_{i=1}^{I}$, i.e., $\mathcal{A}_2Q\mathbf{x}$. This completes the proof. □

Usually we do not know the dimension of the kernel subspace that contains the mapped data samples (including training and test data samples), but given sufficient number of training samples, we can estimate the dimension of this kernel subspace. Assume the rank of the mapped training data samples $\{\varphi(\mathbf{x}_n)\}_{n=1}^{N}$ is N_t, i.e., the mapped training data samples are contained in an N_t-dimensional kernel subspace denoted by $\mathcal{S}$. Suppose the kernel space $\bar{\mathcal{X}}$ has an orthonormal basis $\{\mathbf{e}^{(i)}\}_{i=1}^{\bar{I}}$.

Proposition 1 states that KLFE is basis rotation invariant for bases in the kernel space.

Proposition 1. *Let $\{\mathbf{e}_1^{(i)}\}_{i=1}^{\bar{I}}$ and $\{\mathbf{e}_2^{(i)}\}_{i=1}^{\bar{I}}$ be two orthonormal bases in kernel space. Assume that $\{\mathbf{e}_2^{(i)}\}_{i=1}^{\bar{I}}$ can be obtained by counterclockwise rotating $\{\mathbf{e}_1^{(i)}\}_{i=1}^{\bar{I}}$, and the rotation matrix is denoted by Q. Then, KLFE is basis rotation invariant for all samples $\mathbf{x}$ where $\varphi(\mathbf{x}) \in \mathcal{S}$, i.e., a feature vector extracted by KLFE under $\{\mathbf{e}_1^{(i)}\}_{i=1}^{\bar{I}}$ for input sample $\mathbf{x}$ is the same as the feature vector extracted by KLFE under $\{\mathbf{e}_2^{(i)}\}_{i=1}^{\bar{I}}$.*

Proof. Assume the training sample set is

$$\mathcal{D} \triangleq \{(\mathbf{x}_n, y_n)\}_{n=1}^{N} \subset \mathcal{X} \times \mathcal{Y},$$

where $\mathcal{X} \subset \mathbb{R}^I$ is the I-dimensional feature space and $\mathcal{Y} = \{\pm 1\}$.

In KLFE, a sample is first mapped from a low-dimensional space to a high-dimensional space by the following nonlinear transformation

$$\varphi : \mathcal{X} \subset \mathbb{R}^I \to \bar{\mathcal{X}} \subset \mathbb{R}^{\bar{I}}. \tag{42}$$

The training sample set in the kernel space under basis $\{\mathbf{e}_1^{(i)}\}_{i=1}^{\bar{I}}$ is denoted by

$$\mathcal{D}_1 \triangleq \{(\varphi(\mathbf{x}_n), y_n)\}_{n=1}^{N} \tag{43}$$

Under basis $\{\mathbf{e}_2^{(i)}\}_{i=1}^{\bar{I}}$, the training sample set in the kernel space becomes

$$\mathcal{D}_2 \triangleq \{(Q\varphi(\mathbf{x}_n), y_n)\}_{n=1}^{N}$$

Denote the projection matrix of KLFE in (30) under basis $\{\mathbf{e}_1^{(i)}\}_{i=1}^{\bar{I}}$ and $\{\mathbf{e}_2^{(i)}\}_{i=1}^{\bar{I}}$ by $\bar{\mathcal{A}}_1$ and $\bar{\mathcal{A}}_2$, respectively. Note that the dimension of the feature vector extracted by KLFE is I', where $I' < \bar{I}$. Since KLFE is equivalent to applying LFE in the kernel space, from Lemma 1, we have

$$\bar{\mathcal{A}}_1\varphi(\mathbf{x}) = \bar{\mathcal{A}}_2 Q\varphi(\mathbf{x}) \tag{44}$$

This completes the proof. □

Proposition 2. *Let $\{\mathbf{e}_1^{(i)}\}_{i=1}^{\bar{I}}$ be an orthonormal basis in kernel space. Denote by $\mathcal{S}$ the kernel subspace spanned by training data $\{\varphi(\mathbf{x}_n)\}_{n=1}^{N}$; the dimension of $\mathcal{S}$ is N_t. Then an $\bar{I}$-dimensional* *feature vector $f(x)$ extracted by KLFE under $\{\mathbf{e}_1^{(i)}\}_{i=1}^{\bar{I}}$ for input sample $\mathbf{x}$ (where $\varphi(\mathbf{x}) \in \mathcal{S}$) must be in the form of*

$$f(x) = \left[\begin{array}{c} f_1(x) \\ \mathbf{0}_{\bar{I}-N_t,1} \end{array} \right] \tag{45}$$

where $f_1(x)$ is an N_t-dimensional vector and $\mathbf{0}_{\bar{I}-N_t,1}$ is an $(\bar{I}-N_t)$-dimensional vector whose entries are all zero. In other words, the last $\bar{I}-N_t$ entries of the extracted feature vector $f(x)$ are all zero.

Proof. Assume the training sample set is

$$\mathcal{D} \triangleq \{(\mathbf{x}_n, y_n)\}_{n=1}^{N} \subset \mathcal{X} \times \mathcal{Y},$$

where $\mathcal{X} \subset \mathbb{R}^I$ is the I-dimensional feature space and $\mathcal{Y} = \{\pm 1\}$.

After nonlinear mapping, the training sample set in the kernel space under basis $\{\mathbf{e}_1^{(i)}\}_{i=1}^{\bar{I}}$ becomes

$$\mathcal{D}_1 \triangleq \{(\varphi(\mathbf{x}_n), y_n)\}_{n=1}^{N} \tag{46}$$

Let $\{\mathbf{e}_2^{(i)}\}_{i=1}^{N_t}$ be an orthonormal basis for $\mathcal{S}$. Denote by $\mathcal{S}^{\perp}$ the complementary subspace of $\mathcal{S}$. Let $\{\mathbf{e}_3^{(i)}\}_{i=N_t+1}^{\bar{I}}$ be an orthonormal basis for $\mathcal{S}^{\perp}$. Thus $\{\mathbf{e}_2^{(i)}\}_{i=1}^{N_t} \bigcup \{\mathbf{e}_3^{(i)}\}_{i=N_t+1}^{\bar{I}}$ form an orthonormal basis for kernel space.

From Proposition 1, the feature vector extracted by KLFE algorithm under basis $\{\mathbf{e}_1^{(i)}\}_{i=1}^{\bar{I}}$ is the same as that extracted by KLFE under a rotated basis $\{\mathbf{e}_2^{(i)}\}_{i=1}^{N_t} \bigcup \{\mathbf{e}_3^{(i)}\}_{i=N_t+1}^{\bar{I}}$. So we can compute the extracted feature vector under basis $\{\mathbf{e}_2^{(i)}\}_{i=1}^{N_t} \bigcup \{\mathbf{e}_3^{(i)}\}_{i=N_t+1}^{\bar{I}}$ for simplicity.

Since $\varphi(\mathbf{x}) \in \mathcal{S}$, hence the last $\bar{I}-N_t$ entries of the coordinates of $\varphi(\mathbf{x})$ under basis $\{\mathbf{e}_2^{(i)}\}_{i=1}^{N_t} \bigcup \{\mathbf{e}_3^{(i)}\}_{i=N_t+1}^{\bar{I}}$ are all zero. From the definition of $\bar{\Sigma}_{mh}$ in Eq. (29), we have

$$\bar{\Sigma}_{mh} = \left[\begin{array}{cc} \bar{\Sigma}_{mh}^{*} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{array} \right] \tag{47}$$

where $\bar{\Sigma}_{mh}$ is an $\bar{I} \times \bar{I}$ matrix, and $\bar{\Sigma}_{mh}^{*}$ is an $N_t \times N_t$ matrix.

Let $\bar{\mathcal{A}}$ denote the projection matrix of KLFE in (30) under basis $\{\mathbf{e}_2^{(i)}\}_{i=1}^{N_t} \bigcup \{\mathbf{e}_3^{(i)}\}_{i=N_t+1}^{\bar{I}}$. From Corollary 1.1, Eq. (30) and Eq. (47), we have

$$\bar{\mathcal{A}} = \left[\begin{array}{cc} \bar{\mathcal{A}}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{array} \right] \tag{48}$$

where

$$\bar{\mathcal{A}}_1 = \left[\sqrt{\bar{\sigma}_1}\bar{\mathbf{a}}_1, \ldots, \sqrt{\bar{\sigma}_{N_t}}\bar{\mathbf{a}}_{N_t}\right]^T, \tag{49}$$

and $\{(\bar{\sigma}_i, \bar{\mathbf{a}}_i)\}_{i=1}^{N_t}$ is the eigen-system of $\bar{\Sigma}^*_{mh}$.

From (30), the extracted feature vector $f(x)$ is given by

$$f(x) = \bar{\mathcal{A}}\varphi(\mathbf{x}) \tag{50}$$

$$= \begin{bmatrix} \bar{\mathcal{A}}_1\varphi^*(\mathbf{x}) \\ \mathbf{0} \end{bmatrix} \tag{51}$$

where

$$\varphi(\mathbf{x}) = \begin{bmatrix} \varphi^*(\mathbf{x}) \\ \mathbf{0} \end{bmatrix} \tag{52}$$

This completes the proof. $\square$

It is worth mentioning that if $\varphi(\mathbf{x}) \notin \mathcal{S}$, then the distance $d(\varphi(\mathbf{x}), \varphi'(\mathbf{x}))$ is negligible where $\varphi'(\mathbf{x})$ is the projection of $\varphi(\mathbf{x})$ onto $\mathcal{S}$. The reason is that if the training data $\{\mathbf{x}_n\}_{n=1}^N$ and the test data $\mathbf{x}$ are sampled from the same distribution, $||\varphi(\mathbf{x}) - \varphi'(\mathbf{x})||$ is mainly caused by irrelevant features or measurement noise [10].

From Proposition 2, we can perform feature extraction in kernel subspace $\mathcal{S}$ in which the basis can be expressed by linear combinations of mapped data samples in the kernel space. In this way, we can simplify the computation involved in KLFE. Proposition 2 shows that KLFE can extract at most N_t dimensional nonzero feature vector for arbitrary input sample which lies in $\mathcal{S}$.

Based on the two propositions, KLFE can be computed in three steps. First, we find a basis in kernel subspace. This can be done by using KPCA or Kernel Gram-Schmidt Procedure (KGP) [10]; the dimension of the basis is equal to the rank of the mapped data set in kernel space. Second, the data in the kernel space can be mapped onto the basis, each basis vector of which is a linear combination of the mapped data $\{\varphi(\mathbf{x})\}$, i.e., $v^{(i)} = \sum_j \alpha_{ij}\varphi(\mathbf{x}^{(j)})$. Note that the kernel method can be used to obtain the kernel feature under the basis. Third, we perform LFE on the resulting kernel features which have dimension of N_t.

Next, we present the final KLFE algorithm by using KPCA to find a basis in kernel subspace.

2.4.3 KLFE using KPCA

For a given φ, its kernel function, $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, is defined as

$$\mathcal{K}(\mathbf{x}_1, \mathbf{x}_2) = \langle \varphi(\mathbf{x}_1), \varphi(\mathbf{x}_2) \rangle, \tag{53}$$

where $< \cdot, \cdot >$ represents inner-product operator. It is known that $\mathcal{K}$ and φ have 1-to-1 mapping [11]. In other words, we can ignore the explicit form of φ by using a given $\mathcal{K}$ directly, as long as all computations are conducted through inner product.

Without loss of generality, assume the average of the data samples in kernel space is zero. Let $K \triangleq \bar{X}^T \bar{X}$ be the kernel matrix, where matrix $\bar{X} \triangleq [\varphi(\mathbf{x}_1), \varphi(\mathbf{x}_2), \cdots, \varphi(\mathbf{x}_N)]$. Hence the entry of i-th row, j-th column in K is given by

$$K_{i,j} =< \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) >= \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j), \quad i = 1, \ldots, N; j = 1, \ldots, N. \tag{54}$$

Let $\{\gamma_n, \mathbf{v}_n\}_{n=1}^N$ be the eigen-system of K, where the eigenvalues are sorted in decreasing order, i.e., $\gamma_1 \geqslant \gamma_2 \geqslant \cdots \geqslant \gamma_N$. Then, by definition of eigenvalue decomposition,

$$\bar{X}^T \bar{X} \mathbf{v}_n = \gamma_n \mathbf{v}_n, \tag{55}$$

$$\bar{X}\bar{X}^T \left(\bar{X}\mathbf{v}_n\right) = \gamma_n \left(\bar{X}\mathbf{v}_n\right), \tag{56}$$

for $n = 1, \ldots, N$. As a result, $\{\gamma_n\}_{n=1}^N$ are the N largest eigenvalues of $\bar{X}\bar{X}^T$, whose corresponding eigenvectors are $\{\bar{X}\mathbf{v}_n\}_{n=1}^N$. Denote the dimension of matrix K by N'. If $N' = N$, then $\gamma_i > 0$ for $i = 1, 2, \cdots, N$. If $N' < N$, we perform LFE in the kernel subspace of dimension N'.

Normalizing eigenvectors $\{\bar{X}\mathbf{v}_n\}_{n=1}^N$ produces an orthonormal basis

$$\mathbf{\Psi} = \bar{X}\left[\tfrac{\mathbf{v}_1}{\sqrt{\gamma_1}}, \ldots, \tfrac{\mathbf{v}_N}{\sqrt{\gamma_N}}\right] \triangleq \bar{X}\mathbf{V}'. \tag{57}$$

Thus for input vector $\mathbf{x}_n$ ($n = 1, \ldots, N$), the feature vector extracted by KLFE under the basis in (57) is given by

$$\begin{aligned} \tilde{\mathbf{x}}_n = \; & \mathbf{\Psi}^T \varphi(\mathbf{x}_n) = \begin{bmatrix} \mathbf{v}_1^T/\sqrt{\gamma_1} \\ \vdots \\ \mathbf{v}_N^T/\sqrt{\gamma_N} \end{bmatrix} \bar{X}^T \varphi(\mathbf{x}_n) \\ = \; & \mathbf{V}'^T \bar{X}^T \varphi(\mathbf{x}_n) = \mathbf{V}'^T K^{(n)}, \end{aligned} \tag{58}$$

where $K^{(n)}$ denotes the n-th column of kernel matrix K. More generally, for any $\mathbf{x} \in \mathcal{X}$, we have

$$\tilde{\mathbf{x}} = \mathbf{V}'^T \begin{bmatrix} \mathcal{K}(\mathbf{x}_1, \mathbf{x}) \\ \vdots \\ \mathcal{K}(\mathbf{x}_N, \mathbf{x}) \end{bmatrix} \triangleq \tilde{\varphi}(\mathbf{x}). \tag{59}$$

Eq. (59) actually specifies an N-dimensional kernel space,

$$\tilde{\mathcal{X}} = \{\tilde{\varphi}(\mathbf{x}) : \mathbf{x} \in \mathcal{X}\} \tag{60}$$

We summarize KPCA-based KLFE algorithm as follows. Let

$$\tilde{\mathbf{m}}_n = \tilde{\varphi}(\mathbf{x}_n) - \tilde{\varphi}\left(NM(\mathbf{x}_n, y_n)\right), \tag{61}$$

$$\tilde{\mathbf{h}}_n = \tilde{\varphi}(\mathbf{x}_n) - \tilde{\varphi}\left(NH(\mathbf{x}_n, y_n)\right), \tag{62}$$

$n = 1, \ldots, N$. The KLFE algorithm solves the following optimization problem,

$$\max_{\tilde{\mathbf{W}}} \ \sum_{n=1}^N \tilde{\mathbf{m}}_n^T \tilde{\mathbf{W}} \tilde{\mathbf{m}}_n - \sum_{n=1}^N \tilde{\mathbf{h}}_n^T \tilde{\mathbf{W}} \tilde{\mathbf{h}}_n, \tag{63a}$$

$$\text{s.t.} \quad \|\tilde{\mathbf{W}}\|_F^2 = 1, \tilde{\mathbf{W}} \geqslant 0. \tag{63b}$$

Using the result in Theorem 1, the solution to Eq. (63) is given in Theorem 2.

Theorem 2. *Let*

$$\tilde{\Sigma}_{mh} = \sum_{n=1}^N \tilde{\mathbf{m}}_n \tilde{\mathbf{m}}_n^T - \sum_{n=1}^N \tilde{\mathbf{h}}_n \tilde{\mathbf{h}}_n^T \tag{64}$$

and let $\{(\tilde{\sigma}_i, \tilde{\mathbf{a}}_i)\}_{i=1}^N$ *be the eigen-system of* $\tilde{\Sigma}_{mh}$*, such that* $\tilde{\sigma}_1 \geqslant \cdots \geqslant \tilde{\sigma}_N$. *The solution to Eq.* (63)*, up to the difference of a constant, is*

$$\tilde{\mathbf{W}} = \sum_{\{n:\tilde{\sigma}_n > 0\}} \tilde{\sigma}_n \tilde{\mathbf{a}}_n \tilde{\mathbf{a}}_n^T.$$

Accordingly, the projection matrix is

$$\tilde{\mathcal{A}} = \left[\sqrt{\tilde{\sigma}_1}\tilde{\mathbf{a}}_1, \ldots, \sqrt{\tilde{\sigma}_{I'}}\tilde{\mathbf{a}}_{I'}\right]^T. \tag{65}$$

For an input $\mathbf{x}$*, the extracted feature is given by*

$$f(\mathbf{x}) = \tilde{\mathcal{A}}\tilde{\varphi}(\mathbf{x}) = \tilde{\mathcal{A}}\mathbf{V}'^T \begin{bmatrix} \mathcal{K}(\mathbf{x}_1, \mathbf{x}) \\ \vdots \\ \mathcal{K}(\mathbf{x}_N, \mathbf{x}) \end{bmatrix}. \tag{66}$$

where $\mathbf{V}'$ *is defined in Eq.* (57).

■

KLFE is superior to LFE in that it performs LFE in a high-dimensional space, where discriminant information is much easier to extract. From the above analysis, KLFE can be considered as KPCA followed by LFE. Therefore, KLFE is superior over KPCA since KLFE takes into account the label information; KLFE also outperforms kernel RELIEF, i.e., FSKPCA and FSKGP [10], where FSKPCA is KPCA followed by RELIEF.

2.4.4 KLFE Algorithm

Now the pseudo-code of KLFE is shown. In the initialization step, we need some parameters, like the number of neighbors for computing Σ_{mh}. Assuming that we use the RBF kernel and K-nearest-neighbors as classifier, we need the width of RBF kernel and the number of neighbors for KNN. In our experiments, we use 10-fold cross validation to find these parameters.
If we use complex tuning method to find better parameter set, the performance of KLFE will be improved. In this paper, we do not focus on complex tuning method or classification method. Therefore we use simple tuning method: 10-fold cross validation to tune all parameters needed. For each parameter, we use only 5-10 candidate points.

The complexity of KLFE depends on the kernel function. For example, consider the radial basis function (RBF) kernel [11, page 77], which is given by

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\rho^2}\right). \tag{67}$$

The complexity of computing kernel matrix, i.e., Eq. (54), is $\mathbf{O}(N^2 I)$. The complexity of eigenvalue decomposition for kernel matrix and the complexity of LFE in the N-dimensional kernel space are both $\mathbf{O}(N^3)$. As a result, the overall complexity of KLFE using RBF kernel is

$$\mathbf{O}\left(N^2 I\right) + \mathbf{O}\left(N^3\right). \tag{68}$$

It is comparable to the complexity of LFE, which is also $\mathbf{O}(N^2 I) + \mathbf{O}(N^3)$ [3]. To reduce the computational complexity of KLFE, we can use Kernel Gram-Schmidt Procedure [10] to find a basis instead of using KPCA.

Algorithm KLFE

```
    **Input**: Training samples $X = [x_1 \dots x_N]$ and labels $Y = [y_1 \dots y_N]$
    1) Initialization
        Normalize $X$, give kernel parameter, number of neighbors $L$.
    2) Mapping to kernel space
        2.1) $K = kernel(X)$, $K$ is the kernel of $X$.
        2.2) $[V, D] = EigenDecomposition(K)$,
            $V$'s column contains one principal component, $D$ is a diagonal matrix with eigenvalues.
            All the zero values are removed.
        2.3) $\bar{X} = DV^T K$
    3) LFE
        3.1) for $n = 1 : N$
            $\zeta_i$ is the $i^{th}$ nearest $\bar{x}_i$ labeled the same class with $\bar{x}$
            $\eta_i$ is the $i^{th}$ nearest $\bar{x}_i$ labeled different class with $\bar{x}$
            Then $H_n = [\bar{x}_n - \zeta_1 \dots \bar{x}_n - \zeta_L]$, $M_n = [\bar{x}_n - \eta_1 \dots \bar{x}_n - \zeta_L]$
        3.2) $\Sigma_{mh} = \sum_{n=1}^{N} M_n M_n^T - \sum_{n=1}^{N} H_n H_n^T$
        3.3) $[\bar{V}, \bar{D}] = EigenDecomposition(\Sigma_{mh})$, the same as 2.2.
        3.4) $\tilde{X} = \bar{D}^{1/2} \bar{V}^T \bar{X}$
    4) **Output**: $\tilde{X}$.
```

2.5 Experimental Results

In this section, we conduct experiments on pattern classification to show the performance of our KLFE algorithm and compare it with existing feature extraction schemes. This section is organized as follows. In Section 2.5.1, we describe the experimental setting. In Sections 2.5.2 and 2.5.3, we show the experimental results for simulated data sets and real-world data sets, respectively.

2.5.1 Experimental Setting

We conduct classification experiments on two types of data sets, namely, simulated data sets (sine-surface and Swiss roll) and real-world data sets (UCI Machine Learning Repository [12] and USPS digit handwriting data). In our experiments, we use two data sets from UCI Machine Learning Repository, i.e., data sets for diabetes, and ringnorm. For USPS data, we choose only two digits, namely ”3 versus 5”, since they are the most challenging digits for recognition. (See Fig. 2) We exchange the ”traditional” training sets and testing sets as shown in Table 1.

To make a fair comparison, we compare KLFE with the following feature extraction schemes, which are also based on kernel. We also compare KLFE with origin LFE.

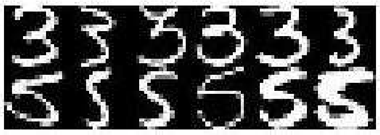


Figure 2: USPS handwritten digits 3(top row) and 5(bottom row).

Table 1: UCI and USPS data sets used in the experiments

data set	training sample size	testing sample size	number of features
UCI-diabetes	468	300	8
UCI-ringnorm	400	7000	20
USPS-(3vs.5)	326	1214	256

1. Generalized Discriminant Analysis (GDA) using a kernel approach [13]
2. KPCA
3. FSKPCA, which is one type of algorithm for kernel RELIEF
4. Kernel K-Nearest Neighbor (KKNN).

GDA can generate at most $n-1$ features where n is the number of categories/classes. In this paper, we only study feature extraction methods for binary classification problem. KKNN is kernelized K-nearest-neighbor (KNN) [6, page 174] based on a distance function induced by the kernel function.

Note that KLFE, GDA, KPCA and FSKPCA are feature extraction algorithms and we are interested in their classification capability, i.e., how well a given classifier performs if the classifier uses the features obtained from these feature extraction algorithms. In our experiments, we choose KNN as the classifier because of two reasons. First, KNN is a simple yet effective classifier, which often yields competitive results, compared to some advanced machine learning algorithms [14]. Second, the focus of this paper is not on an optimal classifier for each dataset. KNN is surely not an optimal classifier in many cases but it provides a platform where we can compare different feature extraction algorithms with a reasonable computational cost. Actually, for fair comparison, we let $K = 1$. Then we can explicitly see how these feature extraction algorithms improve classification ability of KNN in kernel space, i.e., KKNN.

In the experiments, we use RBF kernel function defined by Eq. (67) with $\sigma = 1$. Our comparison strategy is to use the same kernel function with the same width σ and the same classifier KNN where $K = 1$. To eliminate statistical variations, each algorithm is run several times for each data set. In each run, a data set is split into training data subset and testing data subset randomly. Then the testing error rate is obtained by averaging over all the runs.

2.5.2 Experimental Results for Simulated Data

In this section, we conduct experiments on two simulated data sets: twin sine and Swiss roll, which are both in the following form:

$$\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N},$$

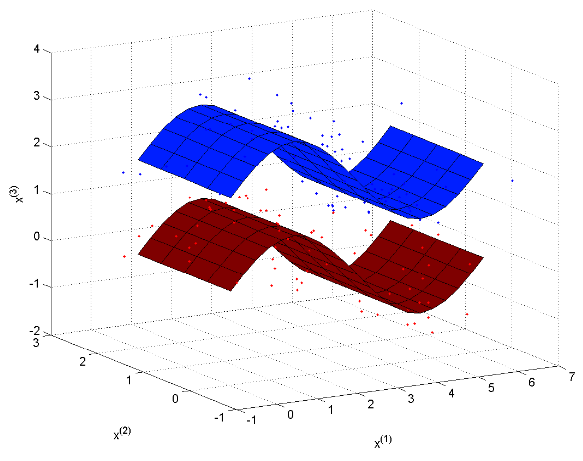
where

$$\begin{aligned} y_n \in & \quad \{-1, 1\}, & (69) \\ \mathbf{x}_n = & \quad R_{I\times 3} \times \begin{bmatrix} \mathbf{x}_n^{(1)} \\ \mathbf{x}_n^{(2)} \\ \mathbf{x}_n^{(3)} \end{bmatrix}, & (70) \\ & & (71) \end{aligned}$$

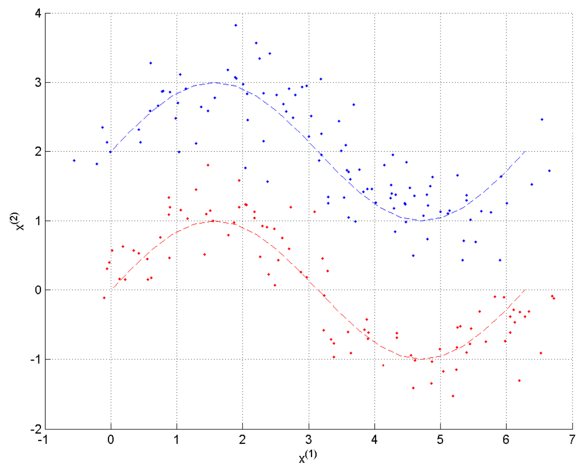
where $R_{I\times 3}$ denotes a random $I \times 3$ matrix.
For twin sine data, $\mathbf{x}_n^{(1)}$ is a random variable uniformly distributed in $[0, 2\pi]$; $\mathbf{x}_n^{(2)}$ is a random variable and $\mathbf{x}_n^{(2)} = \sin\left(\mathbf{x}_n^{(1)}\right) + \mathcal{I}(y_n = 1) \times D + \beta_N$, where $\mathcal{I}(\cdot)$ denotes an indicator function, D is a constant and β_N denotes a Gaussian random variable with zero mean and variance σ^2; $\mathbf{x}_n^{(3)}$ is a random variable uniformly distributed in $[0, 1]$. Actually, the data set is composed of two 3-dimensional sine surfaces, labeled as -1 and 1, with additive Gaussian noise β_N, and the two sine surfaces are separated apart by a distance D. Then the 3-dimensional vector $[\mathbf{x}_n^{(1)}, \mathbf{x}_n^{(2)}, \mathbf{x}_n^{(3)}]^T$ is mapped to a I-dimensional vector by matrix $R_{I\times 3}$.

Fig. 3(a) shows simulated 3-dimensional sine-surfaces and the data set of $[\mathbf{x}_n^{(1)}, \mathbf{x}_n^{(2)}, \mathbf{x}_n^{(3)}]^T$. Fig. 3(b) shows the projection of the data points in Fig. 3(a) onto $\mathbf{x}_n^{(1)} - \mathbf{x}_n^{(2)}$ plane.

We further consider the relationship among classification error rate, separation distance D, and noise variance σ^2. It is obvious that, as D increases, the two sine-surfaces are further away from each other, resulting in better classification accuracy. Similarly, as σ^2 decreases, the probability that samples from two classes overlap decreases, which also increases the classification accuracy.



(a) 3D View



(b) Projection of the data points in Fig. 3(a) onto $\mathbf{x}_n^{(1)} - \mathbf{x}_n^{(2)}$ plane

Figure 3: Simulated data set containing sine surfaces.

Hence, we define signal-noise-rate (SNR) as

$$\text{SNR} = \frac{D^2}{\sigma^2} \tag{72}$$

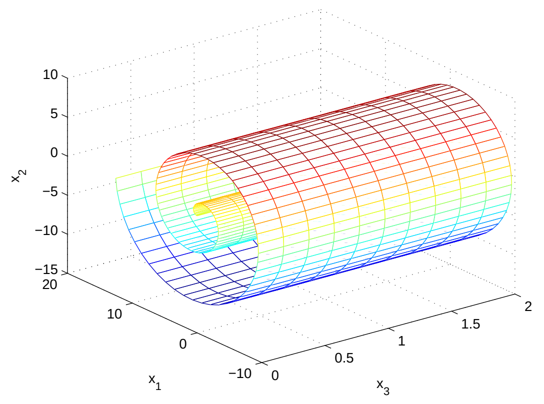
$$\text{SNR (dB)} = 20\log_{10}\left(\frac{D}{\sigma}\right). \tag{73}$$

Figs. 5(a) and 5(b) show classification error rate vs. target feature dimension I' for different schemes under SNR $=$ 0dB and -5dB, respectively.

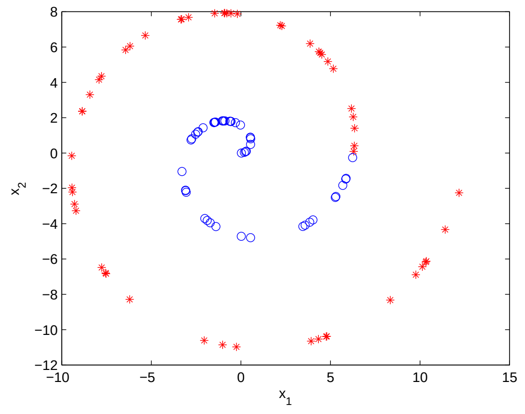
For the Swiss roll data, we let $\mathbf{x}_n^{(1)} = \theta \times \cos(\theta)$ and $\mathbf{x}_n^{(2)} = \theta \times \sin(\theta)$, where θ is a random variable uniformly distributed in $[0, 4\pi]$. Actually the $\mathbf{x}_n^{(1)}$-$\mathbf{x}_n^{(2)}$ curve is a helix. $\mathbf{x}_n^{(3)}$ is a random variable uniformly distributed in $[0, 2]$; then the data set is a 3-D helix surfaces. We label samples with $\theta \in [0, 2\pi]$ as -1 and those with $\theta \in (2\pi, 4\pi]$ as 1. Then the 3-dimensional vector $[\mathbf{x}_n^{(1)}, \mathbf{x}_n^{(2)}, \mathbf{x}_n^{(3)}]^T$ is mapped to a I-dimensional vector by matrix $R_{I\times 3}$.

Fig. 4(a) shows simulated 3-dimensional Swiss roll and the data set of $[\mathbf{x}_n^{(1)}, \mathbf{x}_n^{(2)}, \mathbf{x}_n^{(3)}]^T$. Fig. 3(b) shows the projection of the data points in Fig. 3(a) onto $\mathbf{x}_n^{(1)} - \mathbf{x}_n^{(2)}$ plane.

The classification error rates are all averaged over 10 simulation runs. From Figs. 5(a) and 5(b), it is observed that KLFE+KNN achieves the minimum classification error rate among all the schemes, for $I' \geqslant 10$; KLFE+KNN is able to reduce the classification error rate by more than 10%, compared to other four schemes. From Fig. 6, we can see that KLFE is similar with LFE. Both



(a) 3D View



(b) Projection of the data points in Fig. 4(a) onto $\mathbf{x}_n^{(1)} - \mathbf{x}_n^{(2)}$ plane, where a circle represents a point with label -1, and * represents a point with label 1

Figure 4: Simulated data set containing Swiss roll.

KLFE and LFE are better than PCA. When dimension equals to only 1, KLFE can achieve good results while the other two perform much worse. In addition, our KLFE is quite robust against the change of target feature dimension I'; this is because KLFE has an explicit mechanism to eliminate irrelevant features.

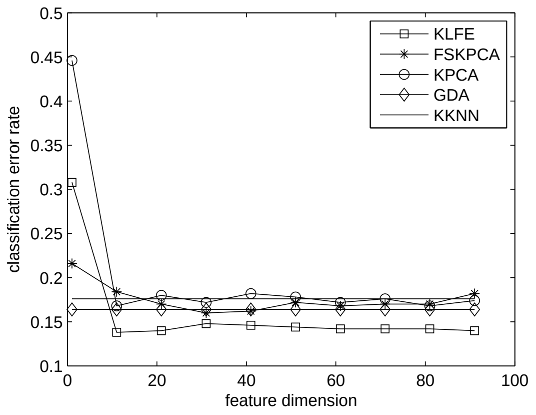
2.5.3 Experimental Results for Real-World Datasets

In this section, we conduct experiments on three real-world data sets: UCI-diabetes, UCI-ringnorm, and USPS-3vs5 (digit “3” vs. “5”).

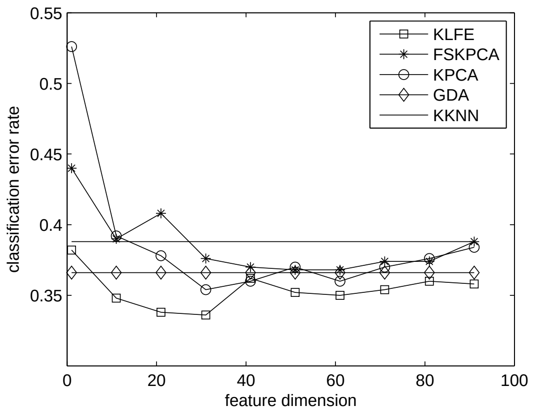
Fig. 7(a) shows classification error rate vs. target feature dimension I' for different schemes on diabetes dataset. It is observed that KLFE+KNN achieves the minimum classification error rate among all the schemes, for $I' \geqslant 30$.

Fig. 7(b) shows classification error rate vs. target feature dimension I' for different schemes on ringnorm dataset. It is observed that KLFE+KNN improves the classification ability of KKNN dramatically, by reducing the classification error rate by more than 50%. GDA also yields good performance but FSKPCA and KPCA give quite poor results, which are even worse than KKNN.

Fig. 8 shows classification error rate vs. target feature dimension I' for three different schemes: KLFE+KNN, LFE+KNN, and PCA+KNN. In this experiment, KLFE performs better than PCA, and achieves performance similar to that of LFE.



(a) TwinSine, SNR $= 0$ dB



(b) TwinSine, SNR $= -5$ dB

Figure 5: Classification error rate vs. target feature dimension of simulated data.

In summary, our proposed KLFE achieves superior performance over the existing algorithms in most cases. Note that there are strong relationship among KPCA, KLFE, and FSKPCA. Both KLFE and FSKPCA find a basis in kernel subspace, differing in that KLFE uses feature extraction matrix to maximize the average margin whereas FSKPCA uses feature weighting to maximize the average margin. Compared to KLFE and FSKPCA, which use supervised learning, KPCA uses unsupervised learning (i.e., without using label information).

It is worth mentioning that our experiments focus on comparison of various feature extraction methods rather than optimal classifier design. In fact, in order to achieve best classification performance using KLFE+KNN, we should select the optimal K for KNN under KLFE, and the best kernel function. But for fair comparison, we just use the same classifier and the same parameter setting for all the feature extraction methods.

2.6 Conclusion

This work is concerned with feature extraction techniques for pattern classification applications. A good feature extraction algorithm is critical in a pattern classification system as it helps reduce system complexity and enhance classification accuracy by eliminating irrelevant features.

In this work, we proposed a novel feature extraction algorithm, referred to as KLFE, which is a generalization of LFE. The power of KLFE lies in the fact that KLFE has the good properties of a feature extraction technique, i.e., it is a nonlinear wrapper feature extraction method that solves a convex optimization problem. Although nonlinearly mapping a pattern to a high-dimensional

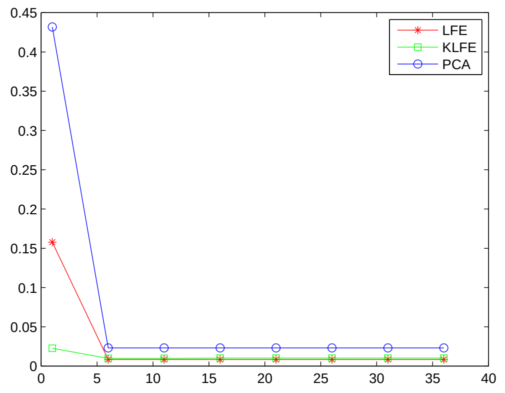
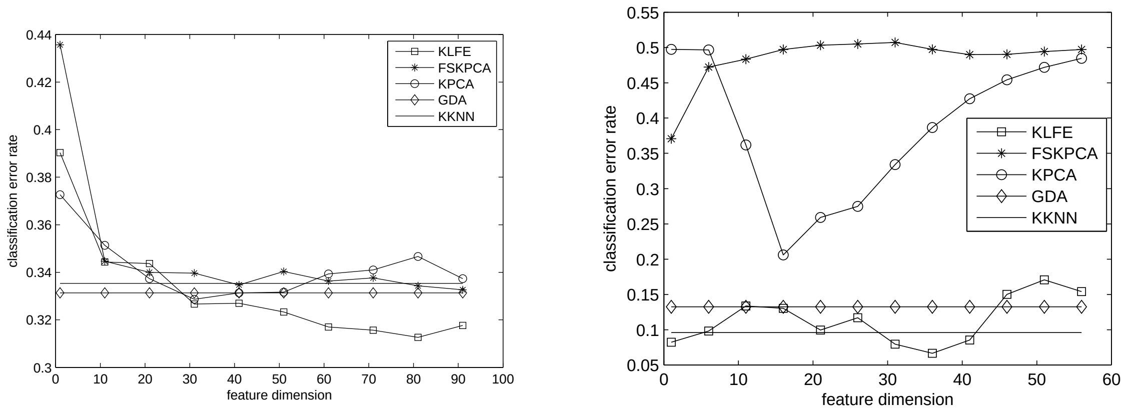


Figure 6: Classification error rate vs. target feature dimension on Swiss Roll



(a) UCI: diabetes

(b) UCI: ringnorm

Figure 7: Classification error rate vs. target feature dimension of UCI data.

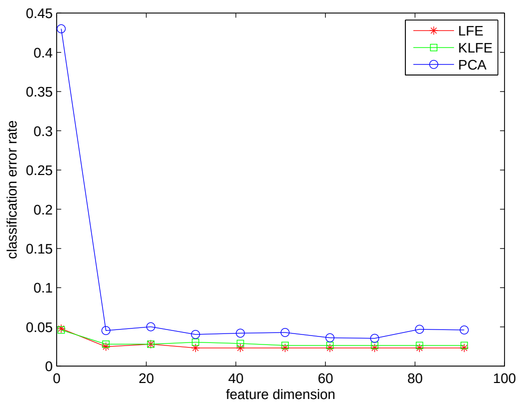


Figure 8: Classification error rate vs. target feature dimension on usps 3 vs. 5.

space followed by LFE, seems to incur extremely high computation complexity, we theoretically proved that LFE and KLFE are both basis rotation invariant, which allows us to implement KLFE via KPCA or KGP followed by LFE.

As shown in Eq. (68), the overall computation complexity of KLFE using RBF kernel function is $\mathbf{O}\left(N^2 I\right) + \mathbf{O}\left(N^3\right)$, comparable to LFE in original feature space. In other words, KLFE has the advantage of better discriminant information extraction in high-dimensional space, while preserving a comparable computation complexity as LFE in low-dimensional space. The experiments conducted on both simulated data set and three real-world data sets demonstrate the effectiveness and robustness of our KLFE algorithm.

3 Identification of Network Dynamics under Sparsity and Stationarity Constraints

The sparse vector autoregressive model (VAR) is commonly used for modeling dynamic networks, such as tank movements, troops movements, brain functional networks, stock markets and social networks. A penalized linear regression was proposed to identify the autoregressive coefficient matrices with sparsity constraint. However, though the VAR model is assumed to be stationary, this property is never taken into consideration by the penalized linear regression. Moreover, the present techniques for estimating a VAR model are only applicable to the problems with relatively low dimensionality and large number of observations. The main purpose of this work is to tackle these challenging issues. We formulate the problem as penalized linear regression with stationarity constraint, and propose the Berhu iterative sparsity pursuit with stationarity constraint (BIPS) to solve the problem efficiently. Berhu is a novel scheme with hybrid penalty that improves the Lasso scheme for high collinearity problem. We also implement the screening technique into BIPS for dealing with the "large p small n" problem. A bootstrap enhanced learning procedure is applied to approximate the probability of existence for each connection. Experiments show that our method guarantees a stationary estimate, outperforms Lasso in estimation accuracy, and works well for high-dimensional problems. Next, we present the technical details.

3.1 Introduction

There is recently much interest in identifying the network dynamics and behaviors in the emerging scientific discipline of network science [15]. In a dynamic network, the evolution of a node is controlled not only by itself, but also by other nodes. Take the gene regulatory network for example, the expression levels of genes interact with each other, following some dynamic rule and structure. The interacting relations connect the genes together, which forms a dynamic network. If the topology and evolution of such network is known, we can analyze the regulation between genes, or detect unusual behaviors to help diagnose and cure genetic diseases. Similarly, the modeling and estimation of dynamic networks is also of great importance for various domains including stock market, brain network [16] and social network [17]. Therefore, to accurately identify the topology and dynamics underlying such network, scientists are devoted to find appropriate mathematical models and corresponding estimation methods.

In practice, we can obtain discrete observations of the network over a period of time. For example, the expression levels of genes are collected at different time points in the microarray experiment, and the macroeconomic data of U.S. are recorded monthly or seasonally. These multivariate time series contains important information for network estimation or analysis. The vector autoregressive model (VAR) is one of the most commonly used models for multivariate time series [18]. In a VAR model, the state of each node in the network is characterized by a time series. The value of a node at the current time point is a linear combination of the past values of itself and other nodes that regulate it. This regulation relationship is illustrated by the model's coefficient matrices, which can be estimated using the observed data.

When estimating the coefficient matrices, we not only want the estimate to fit the training data well, but also need a topology that is easy to interpret and illustrates the most important connections in the network. A "sparse" topology is not only easy to analyze, but also agrees with reality. For example, it has been believed by geneticists that, despite the large scale of a gene regulatory network, each gene is normally only regulated by a few number of other genes [19]. And it is a great challenge to identify such regulation relationships accurately. Compressive sensing approaches such as penalized linear regression are applied by researchers to achieve a small fitting error and a sparse structure simultaneously [20]. Penalized linear regression has been studied a lot in the past decades. Different penalties and algorithms are proposed. A comprehensive review is provided by [21]. The L_1 penalty is popular for its elegance in theory and simplicity in implementation. However, the problem for applying L_1 penalty to estimate the VAR model is its incapability of dealing with collinearity. Moreover, it also suffers from inconsistency and biasness. To improve these drawbacks, we study a new penalty Berhu, which combines the L_1 and L_2 penalties in a novel fashion. An iterative thresholding algorithm is proposed to efficiently solve penalized linear regression with Berhu penalty.

For a network to stay stable and function normally, the VAR model must be stationary. Imagine that the macroeconomic data is not a stationary process, it is easy for one or more indices to grow exponentially, and the whole economic environment must collapse eventually. Mathematically, the maximum likelihood estimation and penalized linear regression all depend on the assumption that the VAR process is stationary [18]. However, stationarity has never been taken into consideration by the estimation algorithms. In fact, as will be shown later in the experiment, it is possible for penalized linear regression to give a nonstationary estimate even when the true model is stationary. In this paper, we aim at dealing with the stationarity property of VAR model, which is of great importance but has never been tackled properly before. We will show that the penalized linear regression method may fail to give a stationary estimate, and propose the Berhu iterative sparsity pursuit with stationarity constraint (BIPS) to overcome this shortcoming. Experiment shows that our method can guarantee a stationary and sparse estimate as well as give a satisfactory identification accuracy. Moreover, when used for forecasting or prediction, our algorithm outperforms the simple penalized regression in a fundamentally different fashion.

Nowadays, we are facing more and more challenges from high dimensional data [22]. For example, A microarray dataset often has thousands of genes but fewer than one hundred of samples. This so-called “large p small n” problem adds difficulties to our estimation method in terms of accuracy as well as computation cost. Dimensionality reduction techniques such as screening [23] can help ease the computation burden and improve accuracy. Nevertheless, a “one-time” estimate, without p-value or confidence interval, may not be satisfactory in practice. To address this issue, we propose the bootstrap enhanced learning procedure. Instead of selecting connections, bootstrap provides us the probability for each connection to exist. We will use the stationary bootstrap [24], which keeps the stationarity property of each bootstrap sample.

This section is organized as follows. Section 3.2 introduces the stationary and sparse VAR model, and formulates an optimization problem to estimate such a model. Section 3.3 proposes our algorithms, mainly the Berhu iterative sparsity pursuit with stationarity constraint (BIPS), and the thresholding-based iterative screening (TIS). The bootstrap enhanced BIPS (BE-BIPS) is described in Section 3.4. Section 3.5 shows experimental results on synthetic data. In Section 3.6, we apply the proposed framework to the U.S. macroeconomic data. Section 3.7 concludes our work.

3.2 The VAR Model and Problem Formulation

In literature, vector autoregressive (VAR) model is commonly used for modeling networks whose dynamics are described by multivariate time series [18]:

$$\boldsymbol{x}_t = \sum_{k=1}^{m} A^{(k)} \boldsymbol{x}_{t-k} + \boldsymbol{\epsilon}_t.$$

In this model, $\boldsymbol{x}$ is a p-dimensional vector with each component being a time series observed from one node in the network, where p is the number of nodes in the network. And $\boldsymbol{\epsilon}_t$ is p-dimensional multivariate Gaussian noise: $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \Sigma_\epsilon)$. The nodes are connected with each other through the autoregressive coefficient matrices $A^{(k)}$'s, and their dynamics are influenced by each other through $A^{(k)}$'s. The VAR model will be generally more powerful in modeling complex dynamic networks with a larger order m. Nevertheless, for most of practical purposes, it is sufficient to use a first-order VAR model to approximate the network behaviors. Hence, we will mainly consider the first-order VAR model:

$$\boldsymbol{x}_t = A\boldsymbol{x}_{t-1} + \boldsymbol{\epsilon}_t, \tag{74}$$

where $A = A^{(1)}$ is the autoregressive coefficient matrix for time lag one. The coefficient matrix $A = \{a_{ij}\}_{1 \leqslant i,j \leqslant p}$ describes a weighted digraph that represents the dynamic network: there is a connection link from node j to node i with weight a_{ij}. We call this graph the connection graph of the network. In econometrics and bioinformatics, this kind of connection is called Granger causality [25], since it illustrates the causal relationship between two nodes.

3.2.1 Conditional Maximum Likelihood Estimation of VAR

Given n observations of the dynamic network $\mathcal{X}$: $\{\boldsymbol{x}_t\}_{t=1}^n$, we wish to infer the connection graph, i.e., to estimate A. Assuming a stationary process, we can write the likelihood function of A as

$$\begin{aligned} L(A|\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n) &= f(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n|A) \\ &= \prod_{t=2}^{n} f(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \cdots, \boldsymbol{x}_2, \boldsymbol{x}_1, A) f(\boldsymbol{x}_1|A) \\ &= \prod_{t=2}^{n} f(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, A) f(\boldsymbol{x}_1|A). \end{aligned}$$

The last but one equation comes from the chain rule of conditional probability. The last equation comes from the Markov-chain property. For simplicity, we consider the conditional likelihood function where $\boldsymbol{x}_1$ is assumed to be not random:

$$\begin{aligned} L_c(A) &= \prod_{t=2}^{n} f(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, A) \\ &= \prod_{t=2}^{n} (2\pi)^{-p/2}|\Sigma_\epsilon|^{-1/2} exp\{-\frac{1}{2}(\boldsymbol{x}_t - A\boldsymbol{x}_{t-1})^T \Sigma_\epsilon^{-1}(\boldsymbol{x}_t - A\boldsymbol{x}_{t-1})\}. \end{aligned}$$

We further assume independent multivariate Gaussian noise with the same variance, which means $\Sigma_\epsilon = \sigma I$. Then the conditional maximum likelihood estimation (CMLE) can be obtained by solving

$$\begin{aligned} \hat{A}_{CMLE} &= \arg\min_A \frac{1}{2}\sum_{t=2}^{n} \|\boldsymbol{x}_t - A\boldsymbol{x}_{t-1}\|_2^2 \\ &= \arg\min_A \frac{1}{2}\|Y - AX\|_F^2 \quad (\triangleq f(A)), \end{aligned} \tag{75}$$

where $Y = [\boldsymbol{x}_2, \boldsymbol{x}_3, \cdots, \boldsymbol{x}_n]$, and $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{n-1}]$. We can see that the conditional maximum likelihood estimate of a VAR model is actually the same with its ordinary linear regression estimate. On the other hand, the exact maximum likelihood estimate requires solving for a nonlinear optimization problem, which is computationally expensive and will not be discussed in detail in this paper.

3.2.2 Penalized Estimation and the Berhu Penalty

The conditional maximum likelihood estimate $\hat{A}_{CMLE}$ is a dense matrix, which corresponds to a complete or nearly complete connection graph. Such a graph is difficult to interpret, and usually does not agree with the reality. As aforementioned, a sparse connection graph is expected for many scenarios. In other words, we are expecting a sparse estimate of A.

The most direct idea to obtain a sparse solution is to add a constraint on $\|A\|_0$ in (75) and solve

$$\begin{aligned} \hat{A} &= \arg\min_A f(A) \\ s.t. &\quad \|A\|_0 \leqslant c, \end{aligned} \tag{76}$$

where $\|A\|_0$ is the L_0 norm of A, which stands for the number of nonzero elements in A, and c is a positive integer that satisfies $c \leqslant p^2$. It is easy to verify that this integer programming problem is NP-hard [26]. Hence, researchers turn to its Lagrange form:

$$\hat{A} = \arg\min_A f(A) + \lambda\|A\|_0.$$

However, due to non-continuity, this problem is still difficult to address. Therefore, researchers propose to use other penalty functions to approximate the L_0. And this leads to the general form of penalized linear regression estimation (PLRE):

$$\hat{A}_{PLRE} = \arg\min_A f(A) + P(A;\lambda). \tag{77}$$

In this paper, we only consider the additive penalties. And for convenience, we denote $P(A;\lambda) = \sum_{i=1}^{p}\sum_{j=1}^{p} P(a_{ij};\lambda_{ij})$, where $P(\bullet)$ is a penalty function applied to each component of the regression coefficient a_{ij}, and λ_{ij} is the corresponding regulation parameter(s). Such a penalty function is designed to enforce a certain kind of sparsity on $\hat{A}$. Hence, (77) aims at obtaining a solution that not only gives a small regression error but also has a sparse structure.

Different penalties have been proposed and studied. The famous Lasso [27] essentially solves the L_1 penalty. It is easy to solve thanks to its convexity. However, Lasso suffers from several drawbacks, such as inconsistency, biasness and incapability of dealing with collinearity. The hard penalty function [28] is designed to mimic the L_0 penalty. It does not introduce bias, but suffers instability in variable selection due to non-continuity. Besides penalty functions with a single thresholding parameter, researchers also propose a variety of hybrid penalties with multiple regulation parameters. Examples include the elastic net [29], the SCAD [30] and the hybrid TISP [31].

In this paper, we are going to adopt a new hybrid penalty Berhu. Berhu gains its name from Huber [32], which is a famous function for robust regression. The Huber function (78) is quadratic in the small variables and linear in the larger ones. When used as a criterion function for regression, the Huber function is only linearly increased by the large errors, which makes it more robust to outliers than the squared-error criterion. Inspired by Huber function, [33] designed a penalty function called "Berhu" (79). As implied by its name, Berhu is a reversed version of Huber: it is linear in small variables and quadratic in larger ones. Figure 9 shows an example of the two functions.

$$H_M(\theta) = \begin{cases} \theta^2 & \text{if } |\theta| \leqslant M \\ 2M|\theta| - M^2 & \text{if } |\theta| > M \end{cases} \tag{78}$$

$$P_B(\theta; \lambda, M) = \begin{cases} \lambda|\theta| & \text{if } |\theta| \leqslant M \\ \lambda\frac{\theta^2+M^2}{2M} & \text{if } |\theta| > M \end{cases} \tag{79}$$

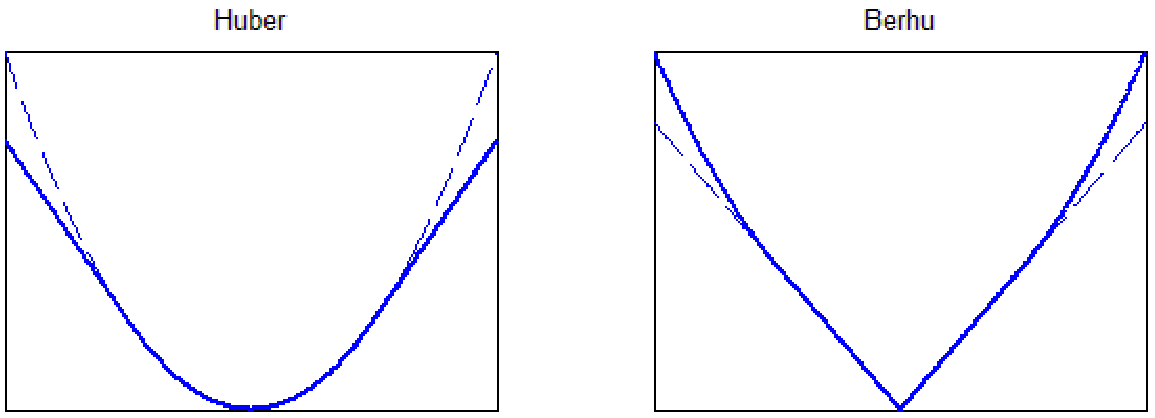


Figure 9: Example of Huber and Berhu.

The Berhu penalty seems quite similar to the elastic net, which also uses both L_1 and L_2 penalties. However, since the elastic net only simply enforces both the L_1 and L_2 penalties simultaneously despite the value of the variable, the singularity of the penalty function at zero is mitigated to some extent by the L_2 part, which may lead to an estimate that is not parsimonious enough. Berhu overcomes this drawback by separating the L_1 and L_2 terms based on the value of the variable. It puts L_1 penalty on the smaller elements, which guarantees sparsity, and puts L_2 penalty on the larger elements, which deals with the collinearity in a similar way with ridge regression [34]. Thus, Berhu not only preserves the singularity property of Lasso at zero but also has the advantages of ridge regression in dealing with high collinearity and high noise level. Compared with SCAD and hybrid TISP, Berhu enjoys favorable properties of convexity. For example, path-wise warm start can be used when tuning regularization parameters. This can save a lot of computation, which makes Berhu more suitable than nonconvex penalties for large-scale problems.

3.2.3 Sparse and Stationary Estimation of the VAR Model

Stationarity is an important property for VAR model. Stationary and nonstationary VAR processes behave fundamentally in different fashions, which can be seen from Figure 10. The probability distribution of observations from a stationary VAR process is invariant with respect to the shift in time. That is, $f(\boldsymbol{x}_{t_1}, \boldsymbol{x}_{t_2}, \cdots, \boldsymbol{x}_{t_n}) = f(\boldsymbol{x}_{t_1+l}, \boldsymbol{x}_{t_2+l}, \cdots, \boldsymbol{x}_{t_n+l})$ for arbitrary $t_1, t_2, \cdots, t_n$, all n and $l = 0, \pm 1, \pm 2, \cdots$. While in the nonstationary process, we can see clearly drifting and trending behaviors. In practice, we often come across stationary VAR processes. Even when the

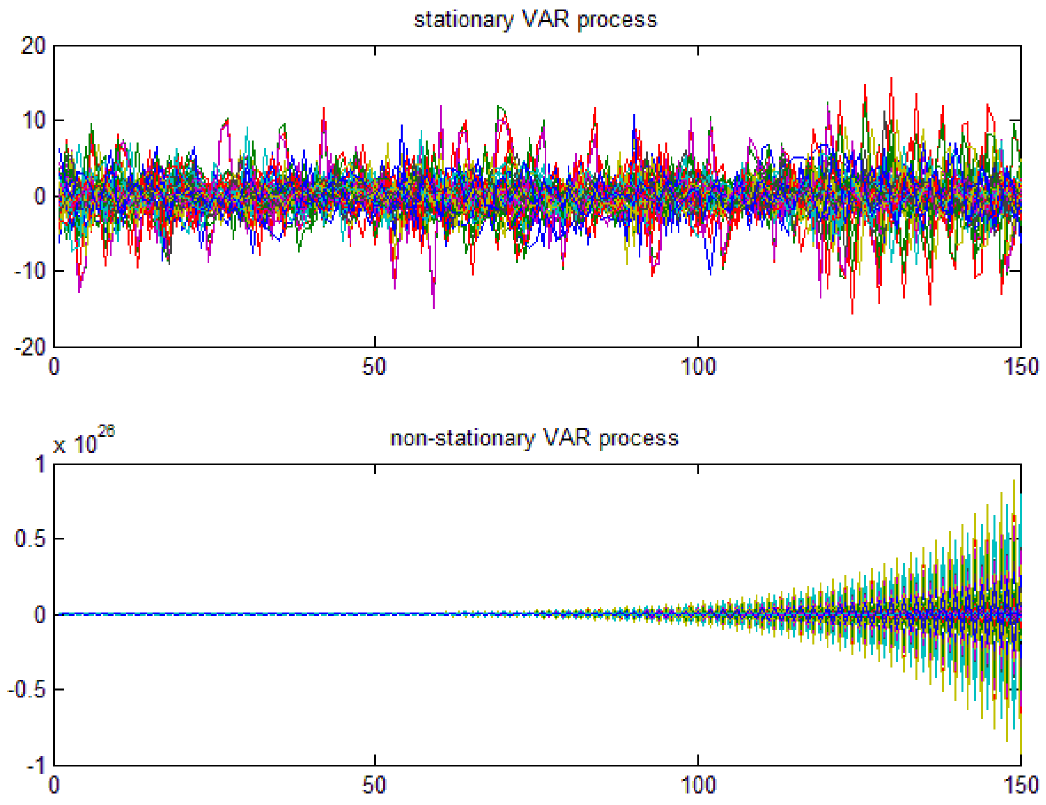


Figure 10: Example of stationary and nonstationary VAR processes. The number of nodes is $p = 50$. The stationary VAR process has $\rho(A) = 0.95$, and the nonstationary VAR process has $\rho(A) = 1.05$.

original process is nonstationary, we can reduce it to a stationary one through some preprocesses, such as differencing the time series and removing the common trend. Therefore, the input to an estimation algorithm is usually a series of stationary observations. As has been shown, the conditional maximum likelihood estimate and the penalized linear regression estimate are all based on the assumption that the VAR process is stationary. So it is natural to require the estimate to also be stationary. However, as will be shown in the experiment, due to estimation bias and error, the penalized linear regression estimate $\hat{A}_{PLRE}$ may violate the stationarity condition. Hence, we need to take stationarity into consideration and reformulate the estimation problem.

For the stationarity property to hold for the VAR model (74), The spectral radius of A should satisfy the *stationarity condition*:

$$\rho(A) \triangleq \max_i |\alpha_i| < 1, \tag{80}$$

where α_i is the ith eigenvalue of A [18]. Therefore, we add the stationarity condition to the penalized linear regression estimation for VAR model:

$$\begin{aligned} \hat{A} = \arg\min_A \ & \frac{1}{2}\|Y - AX\|_F^2 + P_B(A; \lambda, M) \quad (\triangleq l(A)) \\ s.t. \quad & \rho(A) < 1. \end{aligned} \tag{81}$$

In general, the penalty function in (81) can be any penalties we discussed before. We focus on the Berhu penalty $P_B(\bullet)$ just in favor of its advantages in estimating VAR model. The most challenging part of this problem is the constraint on the spectral radius $\rho(\hat{A})$, since $\rho(\hat{A})$ is a nonconvex and non-Lipschitz-continuous function of a generally nonsymmetric matrix A [35]. Hence, we consider a convex relaxation of (80):

$$\|A\|_2 \leqslant 1, \tag{82}$$

where $\|A\|_2$ is the spectral norm (induced 2-norm) of A. Then problem (81) can be reformulated as

$$\begin{aligned} \hat{A} = \arg\min_A \ & l(A) \\ s.t. \quad & \|A\|_2 \leqslant 1. \end{aligned} \tag{83}$$

Throughout this paper, we will call (82) the *stationarity constraint*, and work on solving the above problem.

3.2.4 Equivalent Formulations

Problem (83) is a nonsmooth constrained convex optimization problem, which can be reformulated and then solved by some well-known optimization algorithms, such as semidefinite programming, projected subgradient method and the alternating direction method of multipliers. Before proposing BIPS, we first discuss briefly about these methods.

Semidefinite Programming It can be proved that

$$\|A\|_2 \leqslant 1 \Leftrightarrow \begin{bmatrix} I & A \\ A^T & I \end{bmatrix} \succeq 0.$$

Hence, problem (83) can be reformulated as a semidefinite programming(SDP) problem:

$$\hat{A} = \arg\min_{A} \; l(A)$$

$$s.t. \quad \begin{bmatrix} I & A \\ A^T & I \end{bmatrix} \succeq 0.$$

However, since most of the general SDP solvers use interior point methods, they suffer from very high time complexity and space complexity, especially for large-scale problems [36]. We tried popular SDP solvers SeDuMi [37] and SDPT3 [38] for problem (84) using MATLAB7.11.0 on a PC with 4GB memory. When the network size $p = 100$, the program would run out of memory.

Projected Subgradient Method The projected subgradient method (PSGM) is proposed to solve constrained convex optimization with nonsmooth objective functions [39]. Define the subgradient of the objective function $l(A)$ at A as

$$\partial_\epsilon l(A) = \nabla f(A) + \partial_\epsilon P_B(A; \lambda, M),$$

where $\partial_\epsilon P_B(A; \lambda, M)$ is the ϵ-subdifferential of $P_B(\bullet)$ at A, which is defined by [39]. And $\nabla f(A)$ is the gradient of $f(A)$ at A: $\nabla f(A) = (AX - Y)X^T$. Unlike gradient, which is an exact value for a given point, the subgradient consists of a set of values. Any element in this set is a valid choice for the subgradient.

The PSGM for problem (83) can be described as follows. At each point, we choose a direction U^k from the set of the subgradient $\partial_\epsilon l(A)$, take a step along the opposite direction with a step size α_k, and arrive at a tentative point. Then we project this tentative point to the feasible convex set. We continue like this until reaching a point where 0 is in the subgradient set. The step size α_k should satisfy $\sum_{k=0}^{\infty} \alpha_k = \infty$ and $\sum_{k=0}^{\infty} \alpha_k^2 < \infty$.

For problem (83), the feasible convex set is given by the stationarity constraint:

$$\mathcal{C} = \{A : \|A\|_2 \leqslant 1\}. \tag{84}$$

The projection $\Pi_{\mathcal{C}}(A)$ can be done easily. We take the singular value decomposition (SVD) of A, truncate the singular values that are larger than 1 to 1, and then transfer back to get $\Pi_{\mathcal{C}}(A)$ (**Appendix A**). This operation is called *SVD projection*.

PSGM is simple to implement. However, due to the uncertainness of the subgradient at the non-differentiable point of the penalty function, it suffers from slow convergence and insufficiency of sparsity.

Alternating Direction Method of Multipliers The alternating direction method of multipliers (ADMM) is a method based on the dual ascent method and the augmented lagrangian method [40]. The basic idea is to split the objective function and variables into two parts, and update them in an alternating fashion. To apply ADMM for solving problem (83), we reformulate the problem as

$$\begin{gathered}\min g_1(A) + g_2(B)\\ s.t. \quad A - B = 0\\ A \in \mathcal{C},\end{gathered}$$

where $g_1(A) = \frac{1}{2}\|Y - AX\|_F^2$, and $g_2(B) = P_B(B; \lambda, M)$.

The augmented lagrangian can be written as

$$L_\rho(A, B, \Gamma) = g_1(A) + g_2(B) + \Gamma \circ (A - B) + \frac{\rho}{2}\|A - B\|_F^2, \tag{85}$$

where "$\circ$" denotes the Hadamard product, Γ is the lagrangian multipliers and ρ is the augmented Lagrangian parameter.

The iteration of ADMM for solving (85) consists of the following steps

$$A^{k+1} = \arg\min_{A\in\mathcal{C}} L_\rho(A, B^k, \Gamma^k), \tag{86a}$$

$$B^{k+1} = \arg\min_{B} L_\rho(A^{k+1}, B, \Gamma^k), \tag{86b}$$

$$\Gamma^{k+1} = \Gamma^k + \rho(A^{k+1} - B^{k+1}). \tag{86c}$$

The A-minimization step (86a) can be solved by projected gradient method, which requires inner iterations at each step. The B-minimization step (86b) is solved by subdifferential calculus. According to theoretical analysis and experiment, ADMM also suffers from slow convergence and huge computation if high accuracy is expected.

Having discussed the equivalent formulations and solutions for problem (83), we find a more efficient algorithm is in great need. Therefore, we propose a novel algorithm, the Berhu iterative sparsity pursuit with stationarity constraint (BIPS) in the following section.

3.3 The BIPS Framework

To solve problem (83) efficiently, we first introduce the thresholding rule and iterative solution for Berhu, and then propose the BIPS framework.

3.3.1 The Thresholding Rule for Berhu

Following a three-step procedure [31], we can construct the corresponding thresholding rule of the Berhu penalty (79):

$$T_B(\theta; \lambda, M) = \begin{cases} 0 & \text{if } |\theta| < \lambda \\ \theta - \lambda sgn(\theta) & \text{if } \lambda \leqslant |\theta| \leqslant \lambda + M \\ \frac{\theta}{1+\lambda/M} & \text{if } |\theta| > \lambda + M. \end{cases}$$

It can be seen clearly that L_1 penalty is put on the variables which are less than $\lambda + M$ (the first two cases), and L_2 penalty is put on the variables which are larger than $\lambda + M$ (the third case). We let $\eta = \lambda/M$ and rewrite $T_B(\bullet)$ as

$$T_B(\theta; \lambda, \eta) = \begin{cases} 0 & \text{if } |\theta| < \lambda \\ \theta - \lambda sgn(\theta) & \text{if } \lambda \leqslant |\theta| \leqslant \lambda + \lambda/\eta \\ \frac{\theta}{1+\eta} & \text{if } |\theta| > \lambda + \lambda/\eta. \end{cases}$$

This form of $T_B(\bullet)$ shows that the Berhu penalty does simultaneous selection and shrinkage controlled by a thresholding parameter λ for the L_1 penalty and a ridge parameter η for the L_2 penalty. The threshold for choosing L_1 or L_2 is determined jointly by λ and η. Thanks to the smart combination of the L_1 and L_2 penalties, Berhu not only does selection in a similar fashion with Lasso, but also has the ability to deal with collinearity and high noise level. Figure 11 shows the penalty function and thresholding rule of Berhu, in contrast with Lasso (L_1) and Ridge (L_2) penalty. Note that the soft thresholding corresponds to Lasso. And the ridge thresholding is actually a "shrinking" operation.

Having constructed the thresholding rule for Berhu, we can now use iterative thresholding procedure [41] to solve penalized linear regression (77) with Berhu penalty, which we call the *Berhu sparsity pursuit*:

$$\hat{A}_{Berhu} = \arg\min_A f(A) + P_B(A; \lambda, M). \tag{87}$$

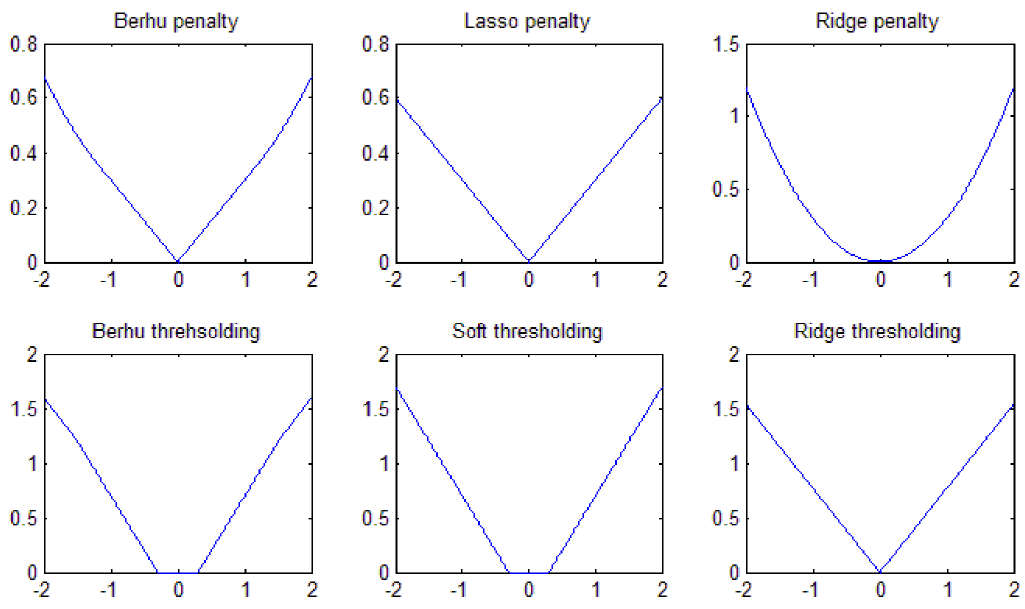


Figure 11: Penalty functions and corresponding thresholding rules. $\lambda = 0.2$, $M = 1.3$.

Starting from an initial estimate, we update the estimate and apply the thresholding rule:

$$A^{k+1} = T_B(A^k + XY^T - XX^TA^k; \lambda, \eta). \tag{88}$$

This procedure goes on iteratively until convergence. Throughout the iteration, some of the entries of the estimate will be shrunk to exactly zero. And the thresholding procedure acts like it is "selecting" the variables that should have nonzero coefficients. Therefore, variable selection and estimation are achieved simultaneously.

It can be proved that, for an arbitrary matrix X, given it is properly preliminarily scaled ($X \leftarrow X/k_0$), the iterative procedure can reach the global minimum of the penalized objective function (**Appendix B**).

It is required that $k_0 \geqslant \|X\|_2/\sqrt{2}$ for convergence guarantee. On the other hand, experience indicates that smaller k_0 leads to faster convergence. Therefore, we choose $k_0 = \|X\|_2/\sqrt{2}$ in practice. Moreover, a relaxation form of the iterative procedure can be used to further speed up the convergence, as given in (89). According to our experience, the number of iterations can be reduced by about 40% compared to the original form if we choose $\omega = 2$.

$$\begin{aligned} \mathscr{A}^{k+1} &= (1-\omega)\mathscr{A}^k + \omega(A^k + XY^T - XX^TA^k), \\ A^{k+1} &= T_B(\mathscr{A}^{k+1}; \lambda) \end{aligned} \tag{89}$$

Unlike PSGM or ADMM, where sparsity and computation speed are hurt by the uncertainness of subdifferentials, the iterative thresholding procedure keeps selecting variables in a determinate fashion, which contributes to fast convergence and sufficient sparsity.

3.3.2 The BIPS Algorithm

Based on the iterative thresholding procedure, we now introduce the BIPS algorithm, where stationarity is addressed. As described by Algorithm 1, the BIPS algorithm consists of two stages. The first stage solves the Berhu sparsity pursuit (87) and gives an estimate $\hat{A}$. We check if the stationarity condition (80) is satisfied by $\hat{A}$. If it is satisfied, we accept and output this solution. Otherwise, we go to the second stage, where we run the iterative procedure again, with a stationarity constraint (82) added in each iteration. That is, at each iteration, we project the updated estimate A^{k+1} onto the convex set (84). Note that X is normalized and preliminarily scaled before running BIPS, so we need to scale A accordingly before and after the SVD projection. This is done by operations $scale$ and $scaleBack$ in Algorithm 1.

In the BIPS algorithm, both the stationarity condition (80) and its convex relaxation, the stationarity constraint (82), is used. In the first stage, we use the stationarity condition to check if the estimate given by Berhu sparsity pursuit is stationary; In the second stage, we consider the stationarity constraint and solve (83), which guarantees a stationary estimate. The reason we keep both stages is that (82) is a sufficient but not necessary condition for (80). Therefore, it may put a too strong constraint on the solution. If we only run the first stage, stationarity is not guaranteed. If we only run the second stage, some estimates that do not violate the stationarity condition may be modified by the stationarity constraint, which leads to a larger estimation error. While by combining the two stages together, BIPS not only guarantees stationarity, but also ensures the identification and estimation accuracy. It may at first sight seem quite time consuming to run two iterative procedures. However, for a stationary model, the probability for the Berhu sparsity pursuit to violate the stationarity condition is actually very small. For most of the cases, the algorithm does not need to run the second stage. Therefore, the average running time of BIPS is in the same order with the Berhu sparsity pursuit.

Algorithm 1 The Berhu iterative sparsity pursuit with stationarity constraint (BIPS)

```
Input: $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{n-1}]$, normalized rowwisely and preliminarily scaled
       $Y = [\boldsymbol{x}_2, \boldsymbol{x}_3, \cdots, \boldsymbol{x}_n]$, centered rowwisely
       regularization parameters: $\lambda$, $\eta$
       relaxation parameter: $\omega$
       stopping criterions: $\delta$, $M$
       initialization for the estimate: $A^0$
   {Definition of two operations: $scale$ and $scaleBack$.
   $scale$: rowwise normalization and preliminary scaling, as has been done to $X$.
   $scaleBack$: inverse operation of $scale$.}
1.
2.  First run:
3.  $k = 0$; $\mathscr{A}^0 = A^0$;
4.  repeat
5.      $\mathscr{A}^{k+1} = (1-\omega)\mathscr{A}^k + \omega(A^k + XY^T - XX^TA^k)$;{update with relaxation}
6.      $A^{k+1} = T_B(\mathscr{A}^{k+1}; \lambda, \eta)$;{thresholding rule for Berhu}
7.      $k = k + 1$;
8.  until $\|A^{k+1} - A^k\|_F \leqslant \delta$ or $k \geqslant M$
9.  $\hat{A} = scaleBack(A^{k+1})$;
10.
11. Check stationarity:
12. if $\rho(\hat{A}) < 1$ then
13.     go to output; {stationarity constraint satisfied. no need for second run}
14. end if
15.
16. Second run:
17. $k = 0$; $\mathscr{A}^0 = A^0 = scale(\hat{A})$;
18. repeat
19.     $\mathscr{A}^{k+1} = (1-\omega)\mathscr{A}^k + \omega(A^k + XY^T - XX^TA^k)$;{update with relaxation}
20.     $A^{k+1} = T_B(\mathscr{A}^{k+1}; \lambda, \eta)$; {thresholding rule for Berhu}
21.     $A^{k+1} = scaleBack(A^{k+1})$;
22.     $A^{k+1} = \Pi_{\mathcal{C}}(A^{k+1})$; {SVD projection}
23.     $A^{k+1} = scale(A^{k+1})$;
24.     $k = k + 1$;
25. until $\|A^{k+1} - A^k\|_F \leqslant \delta$ or $k \geqslant M$
26. $\hat{A} = scaleBack(A^{k+1})$;
27.
Output: $\hat{A}$
```

3.3.3 Thresholding-based Iterative Screening

Nowadays, a great challenge for network inference and statistical learning comes from the large scale of the system, such as world wide web and human genome program. The huge dimensionality p requires the algorithm to have nice scalability. Moreover, we may only be able to obtain just a short snapshot of the system, either because the measurement is too expensive or time consuming, or simply because long time measurement is impossible. Therefore, the "large p small n" problem has become a hot topic [42].

When $p \gg n$, with the assumption that the number of nonzero elements is far smaller than n, we can apply screening techniques to coarsely select the variables before finer estimation. For example, if we are sure that the number of connections for each node is much less than μn (say $\mu = 0.8$), we can first use screening technique to select μn candidate nodes, and then apply BIPS on them for further selection and estimation. Since BIPS is efficient for relatively low dimensional data, we save a lot of time by doing screening first, as long as the screening technique is sufficiently fast and accurate. Here we propose the thresholding-based iterative screening (TIS) for VAR model, given in Algorithm 2.

As implied by its name, TIS depends on an iterative selecting procedure. However, it does not select variables using a given thresholding parameter λ. Instead, TIS keeps a fixed number s ($s = \mu np$) of nonzero elements at each iteration. To be specific, at the $(k+1)$th iteration, we find the sth largest element of the updated estimate A^{k+1}, use it as the current thresholding parameter λ^{k+1}, and apply the thresholding function onto the estimate: $A^{k+1} = T_B(A^{k+1}; \lambda^{k+1}, \eta)$. (Note that though we use Berhu here, the algorithm can be generalized easily to other penalties.) In this way, after each iteration, we obtain an estimate with exactly s nonzero elements, which is the s largest in the updated estimate. Since we only need to obtain the nonzero entries in the screening stage but care little about the exact value of each element, we can choose a relatively large ξ and small M. Also, we can use an empirical value for η, instead of running the algorithm with different values of η and choosing the optimal one. The algorithm's output $S = \{s_{ij}\}_{1 \leqslant i,j \leqslant p}$ is a $p \times p$ matrix recording the entries in $\hat{A}$ that are selected by screening. That is, we have

$$s_{ij} = \begin{cases} 1, & \text{if } \hat{a}_{ij} \neq 0 \\ 0, & \text{if } \hat{a}_{ij} = 0. \end{cases} \tag{90}$$

The sure independent screening (SIS) technique [23] for dimensionality reduction is simple and fast, but it relies on the assumption that the predictors are independent. Since it only calculates

Algorithm 2 Thresholding-based iterative screening (TIS)

```
Input: $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{n-1}]$, normalized rowwisely and preliminarily scaled
       $Y = [\boldsymbol{x}_2, \boldsymbol{x}_3, \cdots, \boldsymbol{x}_n]$, centered rowwisely
       regularization parameters: $\lambda$, $\eta$
       relaxation parameter: $\omega$
       stopping criterions: $\delta$, $M$
       initialization for the estimate: $A^0$
       expected number of nonzeros: $s$
1.
2.  $k = 0$; $\mathscr{A}^0 = A^0$;
3.  repeat
4.      $\mathscr{A}^{k+1} = (1-\omega)\mathscr{A}^k + \omega(A^k + XY^T - XX^T A^k)$;{update with relaxation}
5.      $\lambda^{k+1} = s$th largest element of $\mathscr{A}^{k+1}$; {determine the elements to keep}
6.      $A^{k+1} = T_B(\mathscr{A}^{k+1}; \lambda, \eta)$; {force other elements to be zero}
7.      $k = k + 1$;
8.  until $\|A^{k+1} - A^k\|_F \leqslant \delta$ or $k \geqslant M$
9.  $\hat{A} = A^{k+1}$;
10.
11. let $S$ be a $p \times p$ matrix, and
    $s_{ij} = \begin{cases} 1, & \textit{if } \hat{a}_{ij} \neq 0 \\ 0, & \textit{if } \hat{a}_{ij} = 0 \end{cases}$;
12.
Output: $S$
```

the marginal correlation between Y and X, and chooses the variables accordingly. On the other hand, TIS is more powerful than SIS in dealing with collinearity. This can be seen from the update step of the algorithm, where the information of the correlation matrix XX^T is introduced into the calculation. Although TIS sacrifices some computation time due to the iteration, the great gain in selection accuracy is still worthwhile.

After screening, we can apply BIPS on the reduced model. To achieve this, we just need to add one step in the iteration of BIPS, which confines the updated estimate to the reduced model. That is, after the updating step (line 5 and line 19 in Algorithm 1), we only keep the entries that are selected in the screening stage by letting

$$a_{ij}^{k+1} = \begin{cases} a_{ij}^{k+1}, & \text{if } s_{ij} = 1 \\ 0, & \text{if } s_{ij} = 0. \end{cases}$$

3.3.4 Tuning Strategy

The are two regularization parameters λ and η in BIPS that need tuning. Since the estimate is not quite sensitive to the ridge parameter η, it is not necessary to run a full two-dimensional grid search to look for the best parameters. Instead, we search along a couple of one-dimensional solution paths including the λ-paths (with η fixed) and the η-paths (with λ fixed). Based on our experience, the following "1-3-1" strategy works well:

Step 1: run the first ridge path ($\lambda = 0$). Do ridge regression with different values for the ridge parameter η, and get the optimal ridge parameter η^* as a reference of the λ-paths.

Step 2: run 3 λ-paths with $\eta = 0.5\eta^*, 0.05\eta^*, 0, 005\eta^*$ respectively. For each value of η, run BIPS with a number of values for λ, and find the optimal λ^*. Then we will have three λ^*'s, one from each path. Choose the one that gives the smallest prediction error and let it be the optimal thresholding parameter λ^{**}.

Step 3: run the final η-path with λ^{**}. Choose the optimal ridge parameter η^{**} along the path. The (λ^{**}, η^{**}) is our final choice of the two parameters.

A proper criterion is needed for choosing the optimal parameter at each step. As is well known, if we simply use the squared fitting error to be the criterion, it easily leads to overfitting, especially when the number of observations is limited. Moreover, on the path there are sparse solutions, so we need a criterion that considers different sparsity patterns. For *Step 1*, we use AIC to choose the optimal ridge parameter [43]. For *Step 2* and *Step 3*, we adopt the K-fold selective cross-validation

(SCV) score as the tuning criterion [31]. Here we denote the parameter to be tuned as ω (it can be the thresholding parameter λ or the ridge parameter η). First, we apply the BIPS algorithm to the whole dataset along an ω-path, obtaining the solutions $\hat{A}(\omega)$'s and the associated sparsity patterns $nz_\omega = nz(\hat{A}(\omega))$. Second, we apply K-fold SCV to $\hat{A}(\omega)$ for each ω: for each fold, we run a simple ridge regression on the training data with only the predictors picked by nz_ω, and record the prediction error from the testing data. The averaged prediction error of the K folds is defined to be the SCV score for the corresponding value of ω. Finally, we determine the optimal value ω_{opt} as the one that gives the smallest SCV score along the path.

3.4 Bootstrap Enhanced Learning

The "TIS+BIPS" framework proposed in Section 3 is a powerful tool for sparse and stationary estimation of VAR model in the "large p small n" scenario. However, a "one-time" estimate, without p-value or confidence interval, may not be trustworthy in practice. Bootstrap [44] is a powerful nonparametric tool for approximating the distributions of statistics, confidence intervals, or rejection probabilities of tests. It resamples the data and recalculates the statistics using the resampled data. From the recalculated statistics, we can estimate the distributions of interest and construct confidence intervals. Hence, we propose the bootstrap enhanced BIPS (BE-BIPS), which provides a measure of the confidence about weather a connection exists in the VAR model.

3.4.1 The BE-BIPS Framework

Assume that the connections between two arbitrary nodes in the network follows a distribution that

$$a_{ij} \begin{cases} = 0, \text{with probability } 1 - \xi_{ij} \\ \neq 0, \text{with probability } \xi_{ij}. \end{cases}$$

By bootstrapping, we can approximate this distribution and obtain the empirical value for ξ_{ij}. The BE-BIPS framework is described as follows:

Step 1: run BIPS over the original dataset $\mathcal{X}$. Record the pattern of $\hat{A}$, which is a $p \times p$ matrix P, defined in the same way as S in (90).

Step 2: Draw a bootstrap sample $\mathcal{X}^*$ from the original data. Repeat Step 1 for $\mathcal{X}^*$.

Step 3: Repeat *Step 2* for B times. And record the pattern P_j^* for the jth bootstrap sample.

Adding up all the patterns P_j^*'s and normalizing it by B, we define the matrix $E = \{e_{ij}\}_{1 \leqslant i,j \leqslant p}$ of connection existence probability:

$$E = \frac{1}{B} \sum_B P_j^*.$$

Given a sufficiently large B, the connection existence probability e_{ij} is a good approximation of ξ_{ij}, which can serve as a measure of how confident we are about the existence of each possible connection in the network. For example, if $e_{ij} = 82\%$, it means that in 82% of the bootstrap samples, a connection is identified from node j to node i. So we can say the probability for the existence of this connection is approximately 82%. Given the probability matrix E, we can enforce a threshold e^* on the connection existence probability, and choose only the connections with $e_{ij} \geqslant e^*$ for further study.

The outcome of BIPS can be viewed as a sparse weighted digraph, with the weight of an edge being the regulation strength. On contrast, the outcome of the BE-BIPS is a complete weighted digraph, with the weight of an edge being the probability of its existence.

3.4.2 Stationary Bootstrap

There are different resampling schemes to draw bootstrap samples from the original data in *Step 2*. If the observations are independent and identically distributed, we can resample the data randomly with replacement [44]. When the observations are time series, the problem are more complicated, since the observations are largely dependent on each other, and we would like to keep this dependent information when doing bootstrap. To preserve the temporal dependent structure of the data, techniques such as resampling blocks of consecutive observations or resampling "blocks of blocks" are proposed for bootstrapping time series [45]. The basic idea of block bootstrap methods is that, though individual observations may be dependent, blocks of observations can be approximately independent with each other given a proper block size l.

When the time series is stationary, it is natural to require the pseudo time series obtained by the resampling scheme to be also stationary. The stationary bootstrap [24] is a bootstrap method with this property. It is based on resampling blocks of random length, where the length of each block follows a geometric distribution with mean $1/\tau$. There is a simple method to conduct such resampling. Given that $\boldsymbol{x}_i^*$ is chosen to be the Jth observation $\boldsymbol{x}_J$ in the original time series, we choose $\boldsymbol{x}_{i+1}^*$ based on the following rule:

$$\boldsymbol{x}_{i+1}^* \text{ is } \begin{cases} \text{chosen to be } \boldsymbol{x}_J, \text{with probability } 1-\tau \\ \text{picked randomly from } \{\boldsymbol{x}_t\}_{t=1}^n, \text{with probability } \tau. \end{cases}$$

Similarly with block bootstrap, where the block size l has to be determined, the value of τ should be chosen properly. Good news is that the sensitivity of τ in stationary bootstrap is less than that of l in block bootstrap.

3.5 Experiment

3.5.1 Performance Measures

To examine the performance of the proposed methods, we define the following measures.

Violation Percentage (P_{vio}): In T repeated experiments, if there are T_{vio} experiments in which the estimate $\hat{A}$ violates the stationarity condition (80), then the violation Percentage is defined as $P_{vio} = T_{vio}/T$.

Miss Probability (P_{miss}): If $a_{ij} \neq 0, \hat{a}_{ij} = 0$, we say there is a miss. Denote C_{miss} as the total number of misses and let C_{nz} be the number of nonzero entries in A. The Miss Probability is defined as $P_{miss} = C_{miss}/C_{nz}$.

False Alarm Probability (P_{fa}): If $a_{ij} = 0, \hat{a}_{ij} \neq 0$, we say there is a false alarm. Denote C_{fa} as the total number of false alarms and let C_z be the number of zero entries in A. The false alarm Probability is defined as $P_{fa} = C_{fa}/C_z$.

Prediction Error (prdErr): The prediction error is defined as $prdErr = \|Y_{test}^T - X_{test}^T \hat{A}\|_F / n_{test}$, where Y_{test}^T and X_{test}^T is the testing data, and n_{test} is the length of the testing data.

Running Time (runTime): The averaged running time of an algorithm. All the algorithms are run in MATLAB7.11.0 on a PC with 4GB memory.

3.5.2 Experiment Settings

We generate the $p \times p$ autoregressive coefficient matrix A with both sparsity and stationarity properties. First, the topology is generated from a directed random graph $G(p, \xi)$, where the edge from one node to another node occurs independently with probability ξ. Then, the strength of the edges is generated independently from a Gaussian distribution. This process is repeated until we obtain a matrix A that has the desired spectral radius $\rho(A)$. For all the experiment result shown, $\xi = 0.05, \rho(A) = 0.99$.

For a λ-path, we use a grid of 100 values for λ, which is picked from the interval $[0, \|A^0 + XY^T - XX^TA^0\|_\infty]$. The initial estimate is simply set as $A^0 = \mathbf{0}$. For a η-path, we use a grid of 76 values for η, which is picked from the interval $[2^{-10}, 2^5]$. The number of folds for SCV is set to be $K = 5$. We examine the experiment results for different combination of network size p, sample size n, and noise level σ.

All the statistics we collect are values averaged over 100 repeated experiments.

3.5.3 Performance of BIPS

Table 3.5.3 compares the performance of Lasso, Berhu, and BIPS. Here, by Lasso and Berhu, we mean the penalized linear regression estimates given by the L_1 penalty and the Berhu penalty, respectively.

Comparing the P_{vio}'s, we see that it is possible for the penalized linear regression, no matter Lasso or Berhu is used for penalty, to give a nonstationary estimate for a stationary model. While the proposed BIPS algorithm can guarantee the stationarity property of $\hat{A}$. Therefore, adding the stationarity constraint into the sparsity pursuit does effectively prevent the estimate from becoming nonstationary. Moreover, Berhu outperforms Lasso in both selection accuracy and estimation accuracy, which proves the advantages of Berhu over Lasso. When $p > n$, there is definitely high collinearity in the data matrix X. In this situation, Berhu can do better than Lasso thanks to the L_2 term in the penalty function.

To further illustrate the disadvantages of a nonstationary estimate, we examine its dynamic behavior by looking at the sample paths. For better illustration, we choose a small network size $p = 20$. And $n = 100, \sigma = 1$. From the repeated experiment, we find one run where Lasso gives a nonstationary estimate $\hat{A}_{Lasso}$. Starting from a time point, we observe a sample path from the true model for 100 time points. Then we start from the same initial state and calculate the sample paths for the next 100 time points using both $\hat{A}_{Lasso}$ and $\hat{A}_{BIPS}$. The sample path of the true model and those of the two estimated models are plotted in Figure 12. We can easily see that $\hat{A}_{BIPS}$ gives a reasonable imitation of the true system. However, the nonstationary estimate $\hat{A}_{Lasso}$ blows up quickly and behaves completely different from the true model. This tells us that guaranteeing a

$p/n/\sigma$	algorithm	P_{miss}	P_{fa}	prdErr	P_{vio}
100/50/1	Lasso	0.213	0.139	2.166	0.02
	Berhu	0.185	0.177	2.079	0.04
	BIPS	0.186	0.175	2.075	0
100/50/10	Lasso	0.211	0.135	19.244	0.01
	Berhu	0.188	0.170	19.224	0.02
	BIPS	0.188	0.170	19.223	0
100/80/1	Lasso	0.177	0.189	1.656	0.01
	Berhu	0.157	0.122	1.553	0.01
	BIPS	0.157	0.121	1.553	0
100/80/10	Lasso	0.186	0.194	16.912	0.02
	Berhu	0.169	0.126	15.639	0.03
	BIPS	0.170	0.124	15.623	0

Table 2: Performance comparison of Lasso, Berhu, and BIPS

	p/n=300/80	p/n=400/80	p/n=500/80
TIS	0.229	0.308	0.377
SIS	0.491	0.537	0.579

Table 3: P_{miss} of TIS and SIS

stationary estimate is indeed crucial.

3.5.4 Performance of TIS

To examine TIS's performance for dimensionality reduction and variable selection, we first compare it with the SIS method by looking at the P_{miss} of the two methods. That is, to the same set of data, we apply SIS and TIS separately and compare the patterns obtained by them with the true topology. We let $\mu = 0.9$ in the experiment. Table 3.5.4 shows the result under different (p, n) combinations. We can see that TIS has a much smaller P_{miss} than SIS.

Now we run BIPS with and without TIS and check the difference of the performance. We set the noise level $\sigma = 1$, and compare the two algorithms under different p/n ratios. From table 3.5.4, we can see that, when p/n ratio is large enough, adding TIS not only improves the estimation accuracy, but also saves a lot of time. As the ratio becomes larger, the improvement becomes more significant. On the other hand, when p/n ratio is too large, even TIS fails to be satisfactory. And this is an important motivation for using the bootstrap enhanced learning method.

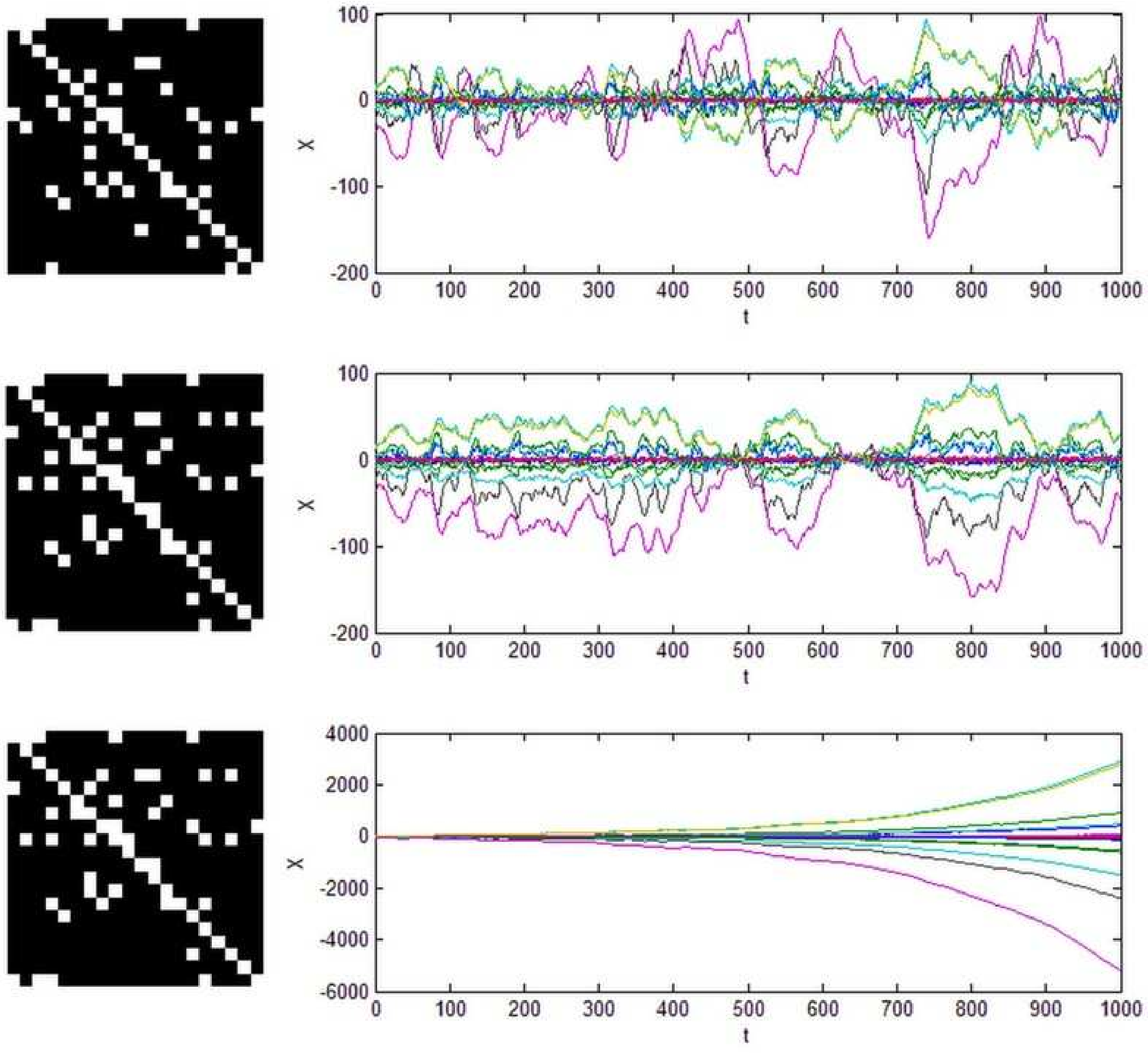


Figure 12: Comparison of Patterns and Sample Paths. Top: pattern of A and observations from the true model; Middle: pattern of $\hat{A}_{BIPS}$ and sample path from the corresponding stationary model; Bottom: pattern of $\hat{A}_{PLASSO}$ and sample path from the corresponding nonstationary model.

3.6 Application to U.S. Macroeconomic Data

We apply the proposed learning framework to the U.S. macroeconomic data. The dataset consists of quarterly observations on 108 macroeconomic variables from 1960:I to 2008:IV, which belong to 12 categories. The dataset with detailed description can be found on "`http://www.`

p/n	algorithm	P_{miss}	P_{fa}	prdErr	runTime
300/80	TIS+BIPS	0.288	0.072	0.985	1216
	BIPS	0.284	0.073	0.987	1363
400/80	TIS+BIPS	0.379	0.058	1.234	1153
	BIPS	0.423	0.050	1.321	3091
500/80	TIS+BIPS	0.473	0.045	1.601	1747
	BIPS	0.529	0.040	1.727	5457

Table 4: Performance of TIS

`princeton.edu/~mwatson/wp.html"`. The data has been preprocessed so that each time series is a stationary process. The stationary and sparse VAR model can be used to identify the Granger causal relationships between different variables.

3.6.1 Comparison of Rolling MSE

First, we use Rolling forecast to compare the performance of Lasso and BIPS. Rolling forecasting procedure [46] is commonly used in macro forecasting. Define the rolling window size to be w. Standing at time point t_0, apply the estimation algorithm to the most recent w observations in the past, i.e., observations from time $t_0 - w + 1$ to t_0: $\{\boldsymbol{x}_t\}_{t=t_0-w+1}^{t_0}$. Then use this estimated model to forecast $\boldsymbol{x}_{t+h}$. This forecasting procedure is repeated as the rolling window slides from the beginning of the time series to the end. Denote the forecast for $\boldsymbol{x}_{t+h}$ as $\hat{\boldsymbol{x}}_{t+h}$. The rolling MSE is defined as $MSE^{rolling} = \frac{1}{n-h-w+1} \sum_{t=w}^{n-h} \|\boldsymbol{x}_{t+h} - \hat{\boldsymbol{x}}_{t+h}\|_2^2$.

We first study the dataset by category. To each of the 12 categories, we apply Lasso and BIPS respectively with $h = 1, w = 0.8 \times d$, where d is the number of time series. Table 3.6.1 shows the rolling MSE of Lasso and BIPS, normalized by that of the AR(4) benchmark, which is a conventional benchmark of macro forecasting. We use the same window size for AR(4) model as the VAR model.

Compared with the AR(4) model, both Lasso and BIPS, which solve a VAR model, have obtained a much smaller forecasting error, except for Category 7. The reason is that, by introducing the Granger causal interactions between different indices, the VAR model becomes more powerful than the univariate AR model in modeling and forecasting, given the same amount of observations. The exception of Category 7 may be due to the higher order of the univariate AR model. Or maybe the Granger causal relationships among Category 7 are weak. The variables are more self-regulated.

Category	1	2	3	4	5	6
Lasso	0.589	0.846	0.936	0.289	0.071	0.506
BIPS	0.445	0.576	0.711	0.165	0.033	0.217
Category	7	8	9	10	11	12
Lasso	1.971	0.552	1.443	0.114	0.370	0.254
BIPS	1.874	0.207	0.738	0.065	0.107	0.100

Table 5: Normalized Rolling MSE of Lasso and BIPS for each category

h	1	2	4	8	16	32
Lasso	0.0174	0.0203	0.0292	0.3647	329.9375	3.1×10^8
BIPS	0.0171	0.0184	0.0187	0.0197	0.0198	0.0176

Table 6: Rolling MSE of Lasso and BIPS for different horizons

Moreover, we note that BIPS gives smaller forecasting errors than Lasso for all the 12 categories of macro time series. It indicates that, by adding a stationarity constraint, we are able to capture the network dynamics more accurately and achieve a stronger capability of forecasting. To further support this conclusion, we apply Lasso and BIPS respectively to the whole dataset, with $w = 0.8 \times d$ and different horizons h. The rolling MSE for $h = 1, 2, 4, 8, 16, 32$ is recorded in Table 3.6.1. As the horizon increases, the rolling MSE of Lasso grows exponentially, which indicates that some estimates of Lasso are nonstationary and therefore completely fail to forecast for large horizons. On the other hand, thanks to the stationarity constraint, the rolling MSE of BIPS stays at the same order of magnitude. This phenomenon is similar with what is shown by Figure 12. They have illustrated the fundamental difference of BIPS from the original penalized linear regression in forecasting capability.

3.6.2 Bootstrap Analysis

We now choose 80 indices for further analysis using stationary bootstrap. We apply the BE-BIPS to these times series and analyze their Granger causal connections. Since the economic structure of U.S. has gone through a big change in the "Great Moderation" in mid-1980 [47], we expect to see significantly different topologies for the macroeconomic network before and after mid-1980. Hence, we divide the time series into two periods, the pre-Great Moderation period and the post-Great Moderation period, and apply BE-BIPS separately to the two periods.

For the pre-Great Moderation period, we choose 80 observations which are from 1960:I to

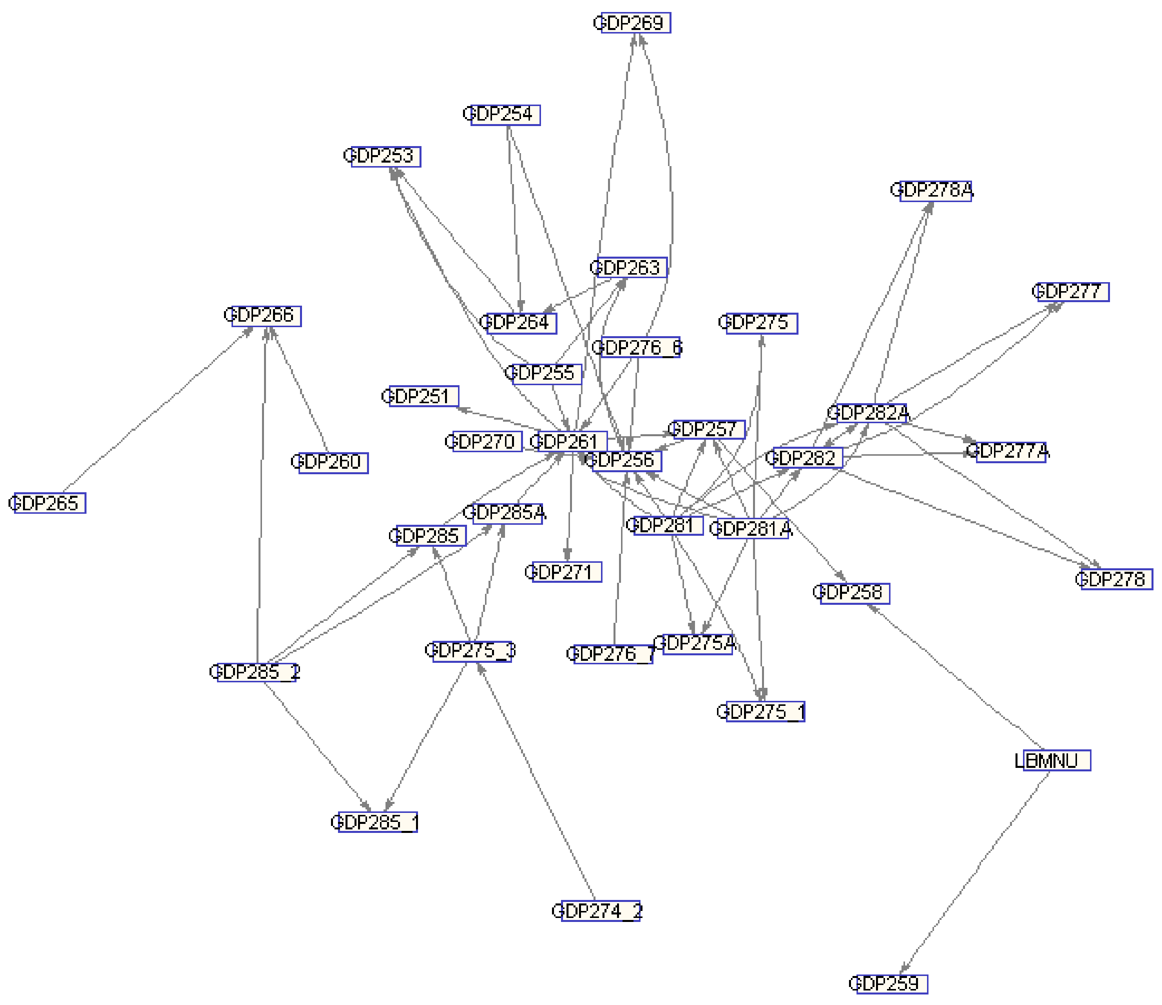


Figure 13: Topology of the macroeconomic network in the pre-Great Moderation period.

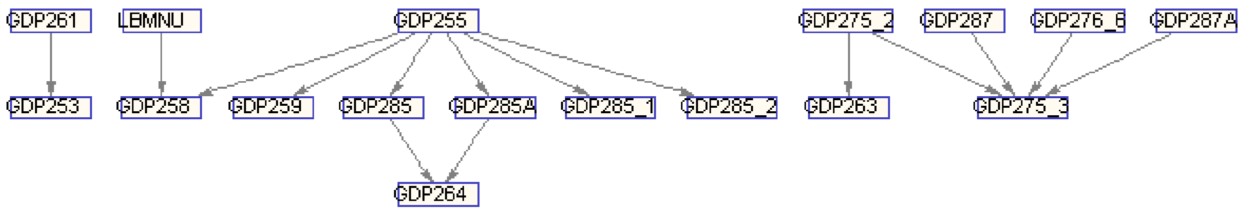


Figure 14: Topology of the macroeconomic network in the post-Great Moderation period.

1979:IV. For the post-Great Moderation period, we choose 80 observations which are from 1985:I to 2004:IV. The time series are resampled using the R function *tsboot* with default parameter values [48]. The number of bootstrap samples is set to be $B = 100$. The threshold for the connection existence probability is chosen to be $e^* = 80\%$. The topologies we obtained for the pre-Great

Moderation period and the post-Great Moderation period are shown in Figure 13 and Figure 14, respectively. In the figures, only the indices that have connection(s) with others are plotted.

After Great Moderation, the business cycle fluctuations have gone through great reduction in volatility. And this reduction has been clearly reflected by the topology change. In the pre-Great Moderation period, the macro variables actively interacted with each other, which forms a very complex dynamic network. While after Great Moderation, the interactions have remarkably reduced, making it easier for the network to stay stable.

3.7 Conclusion

In this paper, we study the estimation of the first order stationary and sparse vector autoregressive (VAR) model. Distinguished from the existing related work, we focus on the stationarity property of VAR and the "large p small n" scenario. To solve the nonstationarity problem of the penalized linear regression estimation, we consider adding a stationarity constraint, and propose the BIPS algorithm to give an efficient solution. For "large p small n" problem, we implement the thresholding-based iterative screening into the BIPS algorithm to improve the identification accuracy and computation efficiency. The bootstrap enhanced BIPS is proposed to provide a confidence measure for each possible connection in the network. The VAR model is widely used in many research domains for time series analysis and network identification. In this paper, we apply it to the analysis of the U.S. macroeconomic data and make some interesting discovery. The proposed framework can also be used to infer the brain function connectivity through fMRI data and the gene regulatory network through microarray data, which will be left for future work.

4 Conclusions

In this project, we have accomplished the research objective of developing transformative theory and algorithms for robust Automated Target Recognition (ATR). Specifically, we have developed the following new techniques:

- kernel local feature extraction (KLFE) for ATR applications,
- technique for identifying network dynamics under sparsity and stationarity constraints,
- self-organized-queue-based (SOQ) clustering scheme,

- robust principal component analysis (RPCA) based on manifold optimization, outlier detection, and subspace decomposition.

References

[1] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *ML92: Proceedings of the ninth international workshop on Machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1992, pp. 249–256.

[2] D. Wettschereck, D. W. Aha, and T. Mohri, "A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms," *Artificial Intelligence Review*, vol. 11, no. 1–5, pp. 273–314, 1997.

[3] Y. Sun and D. Wu, "A relief based feature extraction algorithm," in *Proceedings of SIAM International Conference on Data Mining*, April 2008.

[4] ——, "Feature extraction through local learning," *Statistical Analysis and Data Mining*, vol. 2, no. 1, pp. 34–47, July 2009.

[5] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, pp. 273–324, 1997.

[6] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Wiley-Interscience, Oct. 2000.

[7] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[8] T. G. Dietterich, "Machine-learning research: Four current directions," *AI Magazine*, vol. 18, no. 4, pp. 97–136, Dec. 1997.

[9] Y. Sun and J. Li, "Iterative relief for feature weighting," in *ICML '06: Proceedings of the 23rd international conference on Machine learning*. New York, NY, USA: ACM Press, 2006, pp. 913–920.

[10] B. Cao, D. Shen, J. Sun, Q. Yang, and Z. Chen, "Feature selection in a kernel space," in *Proceedings of the 24th international conference on Machine learning*. ACM New York, NY, USA, 2007, pp. 121–128.

[11] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, June 2004.

[12] D. Newman, S. Hettich, C. Blake, and C. Merz, "UCI repository of machine learning databases," 1998. [Online]. Available: http://www.ics.uci.edu/$\sim$mlearn/MLRepository.html

[13] G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," *Neural computation*, vol. 12, no. 10, pp. 2385–2404, 2000.

[14] B. V. Dasarathy, *Nearest Neighbor: Pattern Classification Techniques*. IEEE Computer Society, Dec. 1990.

[15] M. E. J. Newman and D. J. Watts, *The Structure and Dynamics of Networks. Princeton University Press*. Princeton University Press, 2006.

[16] P. Bellec, V. Perlbarg, S. Jbabdi, M. Pelegrini-Issac, J.-L. Anton, J. Doyon, and H. Benali, "Identification of large-scale networks in the brain using fMRI," *NeuroImage*, vol. 29, no. 4, pp. 1231–1243, Feb. 2006.

[17] T. C. Mills and R. N. MarFkellos, *The Econometric Modelling of Financial Time Series*, 3rd ed. Cambridge University Press, 2008.

[18] G. Reinsel, *Elements of Multivariate Time Series Analysis*, 2nd ed. New York, Springer, 1997.

[19] A. Fujita, J. Sato, H. Garay-Malpartida, R. Yamaguchi, S. Miyano, M. Sogayar, and C. Ferreira, "Modeling gene expression regulatory networks with the sparse vector autoregressive model," *BMC Systems Biology*, vol. 1, no. 1, 2007.

[20] Y. Ren and X. Zhang, "Subset selection for vector autoregressive processes via adaptive lasso," *Statistics and Probability Letters*, vol. 80, no. 23-24, pp. 1705–1712, Dec. 2010.

[21] J. Fan and R. Li, "Statistical challenges with high dimensionality: Feature selection in knowledge discovery," 2006.

[22] I. Fodor, "A survey of dimension reduction techniques," Tech. Rep., 2002.

[23] J. Fan and J. Lv, “Sure independence screening for ultrahigh dimensional feature space,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 70, no. 5, pp. 849–911, 2008.

[24] D. N. Politis and J. P. Romano, “The stationary bootstrap,” *Journal of the American Statistical Association*, vol. 89, no. 428, pp. pp. 1303–1313, 1994.

[25] M. Ding, Y. Chen, and S. L. Bressler, *Granger Causality: Basic Theory and Application to Neuroscience*. Wiley VCH Verlag GmbH Co. KGaA, 2006.

[26] M. Garey and D. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, ser. Series of books in the mathematical sciences. W. H. Freeman, 1979.

[27] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society*, vol. 58, no. 1, pp. 267–288, 1996.

[28] T. Blumensath and M. E. Davies, “Normalized iterative hard thresholding: Guaranteed stability and performance,” *IEEE Journal of selected topics in signal processing*, vol. 4, no. 2, April 2010.

[29] H. Zou and T. Hastie, “Regularization and variable selection via the elastic net,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[30] F. J. and L. R., “Variable selection via nonconcave penalized likelihood and its oracle properties,” *Journal of the American Statistical Association*, vol. 96, pp. 1348–1360, December 2001.

[31] Y. She, “Thresholding-based iterative selection procedures for model selection and shrinkage,” *Electron. J. Statist*, vol. 3, pp. 384–415, 2009.

[32] P. Huber, *Robust statistics*, ser. Wiley series in probability and mathematical statistics. Probability and mathematical statistics. Wiley, 1981.

[33] A. B. Owen, “A robust hybrid of lasso and ridge regression,” Tech. Rep., 2006.

[34] A. T.K. and Wan, “On generalized ridge regression estimators under collinearity and balanced loss,” *Applied Mathematics and Computation*, vol. 129, pp. 455 – 467, Jul 2002.

[35] M. Overton and R. Womersley, “On minimizing the spectral radius of a nonsymmetric matrix function - optimality conditions and duality theory,” *SIAM J. Matrix Anal. Appl.*, vol. 9, no. 4, pp. 473–498, Oct 1988.

[36] H. D. Mittelmann, “An independent benchmarking of sdp and socp solvers.” *Math. Program.*, vol. 95, no. 2, pp. 407–430, 2003.

[37] J. F. Sturm, “Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones,” 1998.

[38] R. H. Ttnc, K. C. Toh, and M. J. Todd, “Solving semidefinite-quadratic-linear programs using sdpt3,” *Mathematical Programming*, vol. 95, pp. 189–217, 2003.

[39] Y. I. Alber, A. N. Iusem, and M. V. Solodov, “On the projected subgradient method for nonsmooth convex optimization in a hilbert space,” *Mathematical Programming*, pp. 23–35, 1998.

[40] S. Boyd, N. Parikh, E. Chu, and J. Eckstein, “Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers,” *Information Systems Journal*, vol. 3, no. 1, pp. 1–118, 2010.

[41] M. Fornasier and H. Rauhut, “Iterative thresholding algorithms,” *Applied and Computational Harmonic Analysis*, vol. 25, no. 2, pp. 187–208, Sept. 2008.

[42] M. Zhang, D. Zhang, and M. Wells, “Variable selection for large p small n regression models with incomplete data: Mapping qtl with epistases,” *BMC Bioinformatics*, vol. 9, no. 1, p. 251, 2008.

[43] K. P. Burnham and D. R. Anderson, “Multimodel inference: understanding AIC and BIC in model selection,” 2004.

[44] B. Efron, “Bootstrap Methods: Another Look at the Jackknife,” *The Annals of Statistics*, vol. 7, no. 1, pp. 1–26, 1979.

[45] H. R. Kunsch, “The Jackknife and the Bootstrap for General Stationary Observations,” *The Annals of Statistics*, vol. 17, no. 3, pp. 1217–1241, 1989.

[46] Makridakis, Wheelwright, and Hyndman, *Forecasting: methods and applications.* Wiley, 1998.

[47] S. J. Davis and J. A. Kahn, “Interpreting the great moderation: Changes in the volatility of economic activity at the macro and micro levels,” National Bureau of Economic Research, Working Paper 14048, May 2008.

[48] P. Dalgaard, *Introductory Statistics with R (Statistics and Computing)*, 2nd ed. Springer, Aug. 2008.